VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
INFORMACINIŲ SISTEMŲ KATEDRA

Tadas Švabauskas

DINAMINĖS KENKSMINGO PROGRAMINIO KODO ELGSENOS RINKIMO PAGERINIMAS PANAUDOJANT VMI

IMPROVING THE COLLECTION OF DYNAMIC BEHAVIOR OF MALICIOUS CODE USING VMI

Baigiamasis magistro darbas

Informacijos ir informacinių technologijų saugos studijų programa,

valstybinis kodas 6211BX014

Informatikos inžinerijos studijų kryptis

Vadovas dr. Dainius Čeponis

Vilnius, 2024

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
INFORMACINIŲ SISTEMŲ KATEDRA

Tadas Švabauskas

DINAMINĖS KENKSMINGO PROGRAMINIO KODO ELGSENOS RINKIMO PAGERINIMAS PANAUDOJANT VMI

IMPROVING THE COLLECTION OF DYNAMIC BEHAVIOR OF MALICIOUS CODE USING VMI

Baigiamasis magistro darbas

Informacijos ir informacinių technologijų saugos studijų programa,

valstybinis kodas 6211BX014

Informatikos inžinerijos studijų kryptis

Vadovas	dr. Dainius Čeponis (Moksl. laipsnis/pedag. vardas, vardas, pavardė)
Konsultantas	_____ (Moksl. laipsnis/pedag. vardas, vardas, pavardė)
Konsultantas	_____ (Moksl. laipsnis/pedag. vardas, vardas, pavardė)
Lietuvių kalbos konsultantas	dr. Vaida Buivydienė (Moksl. laipsnis/pedag. vardas, vardas, pavardė)

Vilnius, 2024

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

INFORMACINIŲ SISTEMŲ KATEDRA

Studijų kryptis: Informatikos inžinerija

Studijų programa: Informacijos ir informacinių technologijų sauga , valstybinis kodas 6211BX014

Specializacija: Informacijos ir informacinių technologijų sauga

TVIRTINU

Katedros vedėjas

Nikolaj Goranin

2024-05-27

MAGISTRO BAIGIAMOJO DARBO UŽDUOTIS

Nr. ITSfm-22–8564

Vilnius

Studentas (-ė): Tadas Švabauskas

Baigiamojo darbo tema: Dinaminės kenksmingo programinio kodo elgsenos rinkimo pagerinimas panaudojant VMI Baigiamojo darbo užbaigimo terminas pagal numatytą studijų kalendorinį grafiką.

BAIGIAMOJO DARBO UŽDUOTIS:

Tikslas: Pagerinti dinaminės kenksmingo programinio kodo elgsenos rinkimą panaudojus VMI techniką.

Uždaviniai:

1. Atlikti dinaminės kenksmingo programinio kodo (KPK) elgsenos duomenų rinkimo būdų analizę.
2. Susipažinti su VMI technikos ypatumais.
3. Atlikti įrankių, kurie naudoja VMI techniką, analizę.
4. Atlikti „Windows“ sisteminių kreipinių rinkimo ir AWSCTD duomenų rinkinio sukūrimo metodo analizę.
5. Pasiūlyti ir praktiškai įgyvendinti pakeitimus, pagerinančius AWSCTD duomenų rinkinio sukūrimą pritaikant VMI techniką „Windows“ sisteminiams kreipiniams rinkti.
6. Atlikti pasiūlyto metodo empirinį tyrimą bei pateikti tyrimo rezultatus ir išvadas.

Planuojamas rezultatas: Dinaminės kenksmingo programinio kodo elgsenos rinkimo pagerinimas panaudojant VMI techniką. AWSCTD duomenų rinkinio papildymas.

Vadovas Dainius Čeponis

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Tadas Švabauskas, 20172794

(Studento vardas ir pavardė, studento pažymėjimo Nr.)

Fundamentinių mokslų fakultetas

(Fakultetas)

Informacijos ir informacinių technologijų sauga, ITSfm-22

(Studijų programa, akademinė grupė)

BAIGIAMOJO DARBO (PROJEKTO)

SĄŽININGUMO DEKLARACIJA

2024 m. gegužės 28 d.

Patvirtinu, kad mano baigiamasis darbas tema „Dinaminės kenksmingo programinio kodo elgsenos rinkimo pagerinimas panaudojant VMI“ yra savarankiškai parašytas. Šiame darbe pateikta medžiaga nėra plagijuota. Tiesiogiai ar netiesiogiai panaudotos kitų šaltinių citatos pažymėtos literatūros nuorodose.

Parenkant ir įvertinant medžiagą bei rengiant baigiamąjį darbą, mane konsultavo mokslininkai ir specialistai: daktaras Dainius Čeponis. Mano darbo vadovas daktaras Dainius Čeponis.

Kitų asmenų indėlio į parengtą baigiamąjį darbą nėra. Jokių įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs (-usi).

Tadas Švabauskas

(Parašas) (Vardas ir pavardė)

Vilniaus Gedimino technikos universitetas

Fundamentinių mokslų fakultetas

Informacinių sistemų katedra

ISBN | ISSN

Egz. sk. .....

Data .....-.....-.....

Antrosios pakopos studijų **Informacijos ir informacinių technologijų saugos** programos magistro baigiamasis darbas

Pavadinimas	**Dinaminės kenksmingo programinio kodo elgsenos rinkimo pagerinimas panaudojant VMI**
Autorius	**Tadas Švabauskas**
Vadovas	**Dainius Čeponis**

Kalba: lietuvių

Anotacija

Tobulėjant technologijai, tobulėja ir nusikaltėlių kuriamos kenkėjiškos programos. Kenkėjiškos programos dažnai taiko antityrimo technikas, kurioms pavaldūs yra tradiciniai („Virtual Machine Introspection“ (VMI) technikos nenaudojantys) dinaminės analizės metodai, todėl atsiranda reikiamybė tobulinti tyrimų metodus taikant pažangesnes technologijas.

Darbo tikslas – pagerinti dinaminės kenksmingo programinio kodo (KPK) elgsenos rinkimą panaudojus VMI techniką. Šio darbo analitinei daliai atlikti buvo naudojama su tema susijusių mokslinių straipsnių bei kitos mokslinės literatūros analizė. Analitinės dalies palyginamoji analizė apėmė „Cuckoo“ ir „Drakvuf-VMI“ technologijų palyginimą, ir buvo nustatyti VMI technikos pranašumai. Išnagrinėtas AWSCTD rinkinio sudarymo metodas, identifikuoti trūkumai bei pasiūlyti patobulinimai. Praktinėje dalyje buvo įgyvendinti dinaminės KPK elgsenos rinkimo patobulinimai, kurių rezultatas – naudojant „Python“ ir „SQLite“ technologijas sukurtas naujasis „Drak Management“ įrankis, kuris automatizuoja procesus, palengvina tyrėjo darbą ir efektyviai renka bei apdoroja duomenis. Empirinis tyrimas, kuriame buvo analizuojama greitaveika ir duomenų generavimas, parodė, kad naudojant VMI techniką, dinaminės KPK elgsenos surinkimas tampa 4–5 kartus efektyvesnis, o sisteminių kreipinių skaičius padidėja maždaug dvigubai, taip praturtinant AWSCTD duomenų rinkinį.

Darbą sudaro 11 dalių: įvadas, dinaminės KPK elgsenos rinkimo metodų analizė, VMI technikos analizė, „Cuckoo“ ir „Drakvuf-VMI“ įrankių analizė bei palyginimas, sisteminių kreipinių rinkimo ir AWSCTD duomenų rinkinio kūrimo metodų analizė, metodo pagerinimo modelio pasiūlymas, dinaminės KPK elgsenos rinkimo pagerinimo įgyvendinimas ir naujojo „Drak Management“ įrankio realizacija, empiriniai VMI patobulinto metodo tyrimai, išvados, literatūros sąrašas ir priedai.

Darbo apimtis – 75 psl. teksto, 42 iliustr., 17 lent., 49 bibliografinių šaltinių ir 8 priedai.

Prasminiai žodžiai: AWSCTD, Virtual Machine Introspection, VMI, dinaminės kenksmingo programinio kodo elgsenos rinkimas, rinkimo pagerinimas, Python, SQLite, Drak Management, Drakvuf, sisteminiai kreipiniai

Vilnius Gediminas Technical University

Faculty of Fundamental Sciences

Department of Information Systems

ISBN | ISSN

Copies No. .....

Date .....-.....-.....

Master Degree Studies **Information and Information Technologies Security** study programme Master Graduation Thesis

Title	**Improving the Collection of Dynamic Behavior of Malicious Code Using VMI**
Author	**Tadas Švabauskas**
Academic supervisor	**Dainius Čeponis**

Thesis language: Lithuanian

Annotation

As technology advances, so does the malware created by bad actors. Malware often employs anti-analysis techniques, which traditional (nonVirtual Machine Introspection (VMI) dynamic analysis methods are susceptible to. Therefore, there is a need to improve malware research methods by leveraging more advanced technologies.

The aim of this work is to improve the collection of dynamic malicious code behavior using VMI technique. The analytical part of this work involved the analysis of scientific articles related to the topic and other scientific literature. The comparative analysis of the analytical part included a comparison of “Cuckoo” and “Drakvuf-VMI” technologies, which resulted in identifying the advantages of VMI technique. The method of composing the AWSCTD dataset was examined, shortcomings were identified, and improvements were proposed. In the practical part, improvements in dynamic malicious code behavior collection were implemented, resulting in the creation of a new tool called “Drak Management” using “Python” and “SQLite” technologies, which automates processes, facilitates the work of the researcher, and effectively collects and processes data. An empirical study, which analyzed performance and data generation, showed that using VMI technique, dynamic malicious code behavior collection becomes 4-5 times more efficient, and the number of system calls increases approximately twofold, thus enriching the AWSCTD dataset.

The work consists of 11 parts: introduction, analysis of dynamic malicious code behavior collection methods, analysis of VMI technique, analysis and comparison of “Cuckoo” and “Drakvuf-VMI” tools, analysis of system call collection and AWSCTD dataset creation methods, proposal of the improvement model, implementation of dynamic malicious code behavior collection improvement and realization of the new “Drak Management” tool, empirical studies of the improved VMI method, conclusions, list of bibliographic references, and appendixes.

The scope of the work is 75 p. of text, 42 illustrations, 17 tables, 49 bibliographic entries & 8 appendixes.

Keywords: AWSCTD, Virtual Machine Introspection, VMI, Malware, collection of dynamic malicious code behavior, improvements, Python, SQLite, Drak Management, Drakvuf, system calls

Turinys

Paveikslų sąrašas 10

Lentelių sąrašas 12

Santrumpų sąrašas 13

Įvadas 14

1. Dinaminės kenksmingo programinio kodo elgsenos rinkimo metodų analizė 18

1.1. Kenkėjiška programinė įranga 18

1.2. Kenksmingo kodo ir programų analizių tipai 20

1.2.1. Statinė analizė 20

1.2.2. Dinaminė analizė 22

1.2.3. Hibridinė analizė 25

1.3. Virtualizacijos metodų iššūkiai 25

1.3.1. „Smėlio dėžių“ veikimo keblumai 25

1.3.2. Virtualizavimo aptikimo technikos, kurias taiko kenkėjiškos programos 25

1.4. Emuliacijos metodas – virtualizacijos alternatyva 27

2. Virtual machine introspection (VMI) analizė 28

2.1.2. Hipervizorių tipai 30

2.1.3. „LibVMI“ 31

2.2. „Drakvuf-VMI“ 32

2.3. „Cuckoo Sandbox“ 33

2.4. „Cuckoo“ ir Drakvuf-VMI palyginimas 35

3. Sisteminių kreipinių duomenų surinkimo metodo tyrimas 39

3.1. DrStrace su ProxMox įgyvendintas metodas 39

3.1.1. Metodo detalizavimas 39

3.1.2. Kenkėjiškų programų paruošimas 40

3.1.3. Metodo rezultatai 44

3.1.4. Metodo pagerinimo potencialas 44

3.2. „AWSCTD“ duomenų rinkinio analizė 45

4. Dinaminės KPK elgsenos rinkimo pagerinimas panaudojant VMI Siūlomas metodo pagerinimo modelis naudojant VMI 47
4.1. „Drakvuf Sandbox“ panaudojimas 48
4.1.1. Aparatinės įrangos bei „Host“ OS reikalavimai 48
4.1.2. „Guest“ VM aplinka 49
4.1.3. „Drakvuf“ konfigūravimas ruošiant daugiau nei vieną „Guest“ VM 51
4.1.4. „Guest“ VM pradinio atvaizdo modifikavimas 52
4.1.5. KPK pavyzdžio analizė 52
4.2. Naujasis „Drak Management“ įrankis 54
4.2.2. Įrankio konfigūracija 56
4.2.3. Įrankio architektūra 60
4.2.1. Duomenų bazių realizacija 64
4.3. Pilnas „Host“ aplinkos paruošimas dinaminės KPK elgsenos rinkimui 67
5. Empiriniai VMI-patobulinto metodo tyrimai 69
5.1. Tyrimui skirtų duomenų atrinkimas 70
5.2. Tyrimui skirtų duomenų paruošimas 70
5.3. Įrangos paruošimas 72
5.3.1. „Host“ aplinkos paruošimas 72
5.3.2. „Guest“ aplinkos paruošimas 73
5.4. Pirmasis empirinio tyrimo etapas 74
5.5. Galutinis empirinio tyrimo etapas 79
Išvados 90
Literatūros sąrašas 91
Priedai 95
Priedas nr. 1 – „Guest“ VM konfigūracijos failas 95
Priedas nr. 2 – „Guest“ VM pradinio atvaizdo modifikacijos 95
Priedas nr. 3 – „Drakvuf Sandbox“ vieno „Windows“ vykdomojo failo pavyzdžio analizės išvesties failų pavyzdys 97
Priedas nr. 4 – „syscall.log“ failo struktūros pavyzdys. 97

Priedas nr. 5 – „Windows 7“ OS „User Account Control Settings“ konfigūravimas 98

Priedas nr. 6 – Sukurtos „Drak Management“ sistemos demonstracija su 100 KPK pavyzdžių imtimi 99

Priedas nr. 7 – 1000 KPK tyrimo rezultatų failai 100

Priedas nr. 8 – Dukteriniai procesai, unikalūs jų pavadinimai gauti atlikus tyrimą su 1000 KPK 100

Paveikslų sąrašas

1.1 pav. KPK paveiktų sistemų augimas („2021 Cyber Security Statistics“, 2021) 19

1.2 pav. Įtariamai kenkėjiškos programos bruožai („Proj 14: Malware Behavior“, n.d.) 22

1.3 pav. „Windows 10“ aplinkoje ‚cache‘ vadinami failai bei katalogai, sukurti po sistemos naudojimo, įdiegus įvairias programas ir jas panaudojus (Buxton, 2021) 23

2.1 pav. VMI architektūros schema (Garfinkel & Rosenblum, 2003) 30

2.2 pav. Hipervizorių tipai 30

2.3 pav. „Drakvuf-VMI“ architektūros schema (Melvin & Kathrine, 2021) 33

2.4 pav. Pagrindinė „Cuckoo“ architektūra (Stichting Cuckoo Foundation, n.d.) 34

2.5 pav. Kenkėjiškų programų skirtingų metodų analizių trukmės (Melvin & Kathrine, 2021)..... 37

2.6 pav. Kenkėjiškų programų pralaidumas (Melvin & Kathrine, 2021)..... 38

3.1 pav. Kenkėjiškų programų vykdymo komponentų schema (Čeponis & Goranin, 2018)..... 40

3.2 pav. Kenkėjiškų programų failų paruošimas (Čeponis & Goranin, 2018) 41

3.3 pav. „VirusShare“ KPK rinkinys..... 42

3.4 pav. „theZoo“ KPK rinkinys 43

3.5 pav. Paruošto KPK rinkinio struktūra 44

3.6 pav. Sistemos kreipinio pavyzdys (Čeponis & Goranin, 2018) 45

3.7 pav. Dažniausiai pasitaikantys sisteminiai kreipiniai (Čeponis & Goranin, 2018)..... 46

4.1 pav. Pasiruošimo planas dinaminės KPK elgsenos rinkimo tyrimui..... 48

4.2 pav. „draksetup install“ „Windows 7“ OS diegimas 50

4.3 pav. „draksetup postinstall“ einamasis procesas po OS diegimo 50

4.4 pav. „Rekall“ profilių gavimas 51

4.5 pav. Architektūrinių sluoksnių diagrama..... 55

4.6 pav. Komponentų tarpusavio sąveikavimo diagrama..... 56

4.7 pav. „Drak Management“ veiklos diagrama (supaprastinta)..... 61

4.8 pav. „Drak Management“ vieno KPK pavyzdžio analizės sąveikavimo su duomenų baze sekų diagrama 62

4.9 pav. „Drak Management“ KPK pavyzdžio analizės suvestinės rašymas į duomenų bazę..... 63

4.10 pav. „Drak Management“ sisteminių kreipinių agregavimo schema (supaprastinta) 64

4.11 pav. Duomenų bazės struktūra 65

5.1 pav. „Virus Share“ bibliotekoje patalpintas archyvas nr. 486, naudojamas šiame tyrime 70

5.2 pav. „VirusShare“ KPK rinkinys 71

5.3 pav. KPK pavyzdžių tyrimų trukmės 75

5.4 pav. Nefiltruotų ir filtruotų sisteminių kreipinių dažniai 77

5.5 pav. 40 daugiausiai surinktų sisteminių kreipinių dažnumas 78

5.6 pav. Sisteminių kreipinių duomenų bazėje pavyzdys 79

5.7 pav. Windows OS rinkos pasiskirstymo grafikas nuo 2017-07 iki 2024-04 (StatCounter, 2024) 82

5.8 pav. Galutinio etapo KPK pavyzdžių tyrimų trukmės 82

5.9 pav. Galutinio etapo KPK pavyzdžių tyrimų trukmės nuo ilgiausios iki trumpiausios 83

5.10 pav. Galutinio etapo daugiausiai kartų aptikti KPK pavyzdžio sisteminiai kreipiniai 84

5.11 pav. Galutinio etapo „Session“ duomenų bazės pavyzdys 84

5.12 pav. Galutinio etapo nefiltruotų ir filtruotų sisteminių kreipinių dažniai 85

5.13 pav. „Drak Management“ bandymų išanalizuoti KPK pavyzdį kiekis 86

5.14 pav. Didžiausias kiekis sisteminio kreipinio perduodamų argumentų 87

5.15 pav. Galutinio etapo 40 daugiausiai surinktų sisteminių kreipinių dažnumas 88

Lentelių sąrašas

2.1 lentelė. „Cuckoo“ ir „Drakvuf“ palyginimas (Melvin & Kathrine, 2021) 35

2.2 lentelė. Kenkėjiškų programų skirtingų metodų analizių trukmės (Melvin & Kathrine, 2021).. 37

2.3 lentelė. Kenkėjiškų programų pralaidumas (Melvin & Kathrine, 2021)..... 38

2.4 lentelė. Vietos diske naudojimas su atminties išmetimu ir be jo (Melvin & Kathrine, 2021) 39

4.1 lentelė. „syscall.log“ sisteminio kreipinio įrašo objektų paaiškinimai..... 53

4.2 lentelė. „analysis_sample“ duomenų bazės struktūros lentelė 65

4.3 lentelė. „syscall_aggregate“ duomenų bazės struktūros lentelė..... 66

4.4 lentelė. „session_log“ duomenų bazės struktūros lentelė..... 66

5.1 lentelė. Windows 7 atvaizdo detalizavimas..... 73

5.2 lentelė. Greitaveikos ir pralaidumo KPK pavyzdžių tyrimo statistinė lentelė 76

5.3 lentelė. Greitaveikos ir pralaidumo tyrimo metu surinktų sisteminių kreipinių statistinė lentelė77

5.4 lentelė. Galutinio empirinio tyrimo etapo suvestinė lentelė..... 81

5.5 lentelė. Galutinio etapo greitaveikos ir pralaidumo KPK pavyzdžių tyrimo statistinė lentelė ... 83

5.6 lentelė. Galutinio etapo tyrimo metu surinktų sisteminių kreipinių statistinė lentelė 85

5.7 lentelė. Galutinio etapo tyrimo metu surinktų vieno KPK pavyzdžio analizės sisteminių kreipinių kiekio statistinė lentelė..... 85

5.8 lentelė. Galutinio tyrimo etapo metu labiausiai su KPK asocijuojamų išgautų kreipinių kiekiai89

5.9 lentelė. Šio empirinio tyrimo rezultatų ir AWSCTD kūrimo metodo rezultatų palyginimas 89

Santrumpų sąrašas

API ***A**pplication **P**rogramming **I**nterface* – taikomojo programavimo sąsaja, užtikrinti skirtingų sistemų tarpusavio komunikacijos ryšį.

AWSCTD ***A**ttack-Caused „**W**indows“ **S**ystem **C**alls **T**races **D**ataset*. Kenkėjiško programinio kodo atakų sukeltų „Windows“ OS sistemos kreipinių pėdsakų duomenų rinkinys.

BIOS ***B**asic **I**nput/**O**utput **S**ystem*. Pagrindinė įvesties/išvesties sistema.

CFG ***C**ontrol **F**low **G**raph*. Proceso diagrama.

DOS ***D**isk **O**perating **S**ystem*. Grupė operacinių sistemų, susidedančių iš paprasto funkcijų rinkinio.

IDS ***I**ntrusion **D**etection **S**ystem*. Įsibrovimų aptikimo sistema.

IDT ***I**nterrupt **D**escriptor **T**able*. Pertraukimo deskriptorių lentelė.

KPK **K**enksmingas **P**rogramins **K**odas

MAC ***M**edia **A**ccess **C**ontrol (address)*. Medijos prieigos valdymas.

OS ***O**perating **S**ystem*. Operacinė sistema.

PE ***P**ortable **E**xecutable file*. Vykdomasis failas.

System calls Sistemos kreipiniai / sisteminiai kreipiniai / sistemos funkcijos – suteikia esminę sąsają tarp proceso ir operacinės sistemos.

VM ***V**irtual **M**achine*. Virtuali mašina.

VMI ***V**irtual **M**achine **I**ntrospection*. Virtualios mašinos introspekcija.

VMM **V**irtualios **m**ašinos **m**onitorius

WMI ***„W**indows“ **M**anagement **I**nstrumentation*. „Windows“ valdymo instrumentai.

Įvadas

Kenkėjiškos programos (virusai) yra naudojami kaip įrankiai atlikti nusikalstamą veiklą kibernetinėje erdvėje. Šios programos varijuoja nuo labai paprastų (pavyzdžiui, programa, kuri ištrina sistemos vartotojo failus) iki sudėtingų (pavyzdžiui, programa, kuri vykdo sistemos vartotojo šnipinėjimą, ar naudoja sistemos resursus kriptovaliutai kasti) programų, galimai naudojančių pažangų šifravimą arba galinčių turėti daug komponentų modulių, kurie praplečia kenkėjiškas funkcijas. Neužtenka aptikti ir pašalinti kenkėjiškų programų artefaktus – labai svarbu suprasti, kaip įvairios kenkėjiškos programos veikia, saugumo pažeidimo kontekstą, motyvus bei tikslus. Tobulėjant technologijai, tobulėja ir nusikaltėlių kuriamos kenkėjiškos programos, todėl yra svarbu jas dinamiškai analizuoti ir suprasti, kaip galėtume sumažinti žalos riziką žmonėms ir verslams.

Kenkėjiškų programų bei jų kodo analizė yra poveikio bei tikslo vienaip ar kitaip paveikti, sukompromituoti su tikslu pakenkti, supratimo procesas. Analizės rezultatai padeda aptikti ir sušvelninti galimą grėsmę. Analizė yra skirstoma į statinę ir dinaminę. Statinės analizės metu programa nėra paleidžiama, o analizės procesus sudaro programos struktūros, jos kodo, funkcijų nagrinėjimai. Atliekamas programos kodo tyrimas, kuriuo metu ieškomi pėdsakai, turintys kenkėjiškų požymių. Tai gali būti naudinga identifikuojant kenkėjišką tinklo infrastruktūrą, sistemines bibliotekas ar supakuotus failus. Dinaminė kenkėjiškų programų analizė vykdo įtariamą kenkėjišką kodą saugioje aplinkoje, vadinamoje „smėlio dėže“ (angl. *sandbox*). Ši kontroliuojama bei izoliuota aplinka suteikia galimybę saugos specialistams stebėti, kaip veikia kenkėjiška programa bei užkerta kelią jai patekti į išorines sistemas, į įmonės tinklą bei plisti per kitus įrenginius. Norint išnaudoti izoliuotos testavimo aplinkos potencialą, kuriami bei tobulinami specialūs, analizei skirti, įrankiai. Dažniausiai, dinaminė analizė (programos vykdymas bei duomenų rinkimas) vyksta izoliuotos aplinkos viduje. Dėl to iškyla kelios problemos: „smėlio dėžės“ aplinka gali būti aptinkama kenkėjiškų programų, analizės duomenys bet kuriuo metu gali būti pažeisti

Šioms problemoms spręsti buvo sukurta virtualios mašinos introspekcijos (VMI) technika, kuri yra naudojama virtualios mašinos (VM) atminčiai ir būsenai analizuoti realiuoju laiku. Taikant šią techniką, kenkėjiška programa gali nežinoti, kad ji yra testuojama specialioje aplinkoje, todėl ir nuslėpti savo kenkėjiško funkcionalumo jai nėra būtina. Analizės duomenys nėra renkami izoliuotos aplinkos viduje, todėl kiekviena sėkmingai paleista programa yra 100 % išanalizuojama. Taigi taikant VMI galima ir patobulinti esamus duomenų rinkimo metodus, kadangi VMI yra nesunkiai prieinama, dažniausiai atvirojo kodo bei daugiamodulinė technologija.

Šiame darbe stengiamasi atsakyti į klausimą, kokius pranašumus atneša VMI technika, taikant ją dinaminėje analizėje bei kaip patobulinti kenksmingo programinio kodo dinaminės elgsenos surinkimą pritaikant VMI.

Tyrimo objektas – metodai, skirti rinkti kenksmingo programinio kodo sisteminių funkcijų sekas.

Darbo tikslas – pagerinti dinaminės kenksmingo programinio kodo elgsenos rinkimą panaudojus VMI techniką.

Darbo uždaviniai:

1. Atlikti dinaminės kenksmingo programinio kodo (KPK) elgsenos duomenų rinkimo būdų analizę.
2. Susipažinti su VMI technikos ypatumais.
3. Atlikti įrankių, kurie naudoja VMI techniką, analizę.
4. Atlikti „Windows“ sisteminių kreipinių rinkimo ir AWSCTD duomenų rinkinio sukūrimo metodo analizę.
5. Pasiūlyti ir praktiškai įgyvendinti pakeitimus, pagerinančius AWSCTD duomenų rinkinio sukūrimą pritaikant VMI techniką „Windows“ sisteminiams kreipiniams rinkti.
6. Atlikti pasiūlyto metodo empirinį tyrimą bei pateikti tyrimo rezultatus ir išvadas.

Temos naujumas – tobulėjant technologijai, tobulėja ir nusikaltėlių kuriamos kenkėjiškos programos. Kenkėjiškos programos dažnai taiko antityrimo technikas, kurioms pavaldūs yra tradiciniai (VMI technikos nenaudojantys) dinaminės analizės metodai, todėl atsiranda reikiamybė tobulinti tyrimų metodus taikant pažangesnes technologijas.

Temos aktualumas – VMI technika suteikia didelį metodų patobulinimų potencialą, kuris padės ateityje paruošti didesnį ir tikslesnį išanalizuotų KPK duomenų rinkinį, skirtą mašininiam mokymui. Šiame darbe įgyvendintas dinaminės KPK elgsenos rinkimo pagerinimas pasitarnaus būsimiems automatizuotiems KPK „Windows“ sisteminių kreipinių rinkimo tyrimams.

Tyrimo metodika – šio darbo analitinei daliai atlikti buvo naudojama su tema susijusių mokslinių straipsnių ir kitos mokslinės literatūros analizė. Analitinėje dalyje buvo lyginamos „Cuckoo“ ir „Drakvuf-VMI“ technologijos bei nagrinėtas AWSCTD rinkinio sudarymo metodas. Praktinėje dalyje buvo patobulintas dinaminės KPK elgsenos rinkimas, sukurtas naujasis „Drak Management“ įrankis, naudojant „Python“ ir „SQLite“ technologijas, bei atliktas empirinis tyrimas, apimantis greitaveikos ir duomenų generavimo analizes.

Mokslinė darbo vertė – pasiūlytas ir įgyvendintas „Windows“ sisteminių kreipinių duomenų surinkimo patobulinimas pritaikant VMI techniką bei atliktas tyrimas, kurio metu gauti rezultatai, galintys papildyti AWSCTD duomenų rinkinį.

Darbo rezultatai:

1. Atlikta literatūros šaltinių, susijusių su KPK tipais, statinės bei dinaminės KPK elgsenos rinkimo metodų analize, įvairių „smėlio dėžių“ architektūromis, VMI technika bei įrankiais, analizė.

2. Išnagrinėtas D. Čeponio „Windows” sisteminių kreipinių rinkimo bei AWSCTD duomenų rinkinio sukūrimo metodas.

3. Pristatytas ir įgyvendintas dinaminės KPK elgsenos surinkimo patobulinimas taikant VMI techniką.

4. Parinktas ir paruoštas KPK bandinių rinkinys, skirtas pagrindiniam empiriniam tyrimui.

5. Atliktas patobulinto metodo empirinis tyrimas siekiant nustatyti pagrindinį jo funkcionalumo korektiškumą.

Darbo struktūra:

Darbą sudaro šeši skyriai, tai: įvadas, dinaminės KPK elgsenos rinkimo metodų analizė, VMI analizė, sisteminių kreipinių rinkimo ir AWSCTD duomenų rinkinio kūrimo metodų analizė, metodo pagerinimo modelio pasiūlymas ir įgyvendinimas, empiriniai VMI patobulinto metodo tyrimai.

Darbo aprobacija:



VILNIUS TECH
Fundamentinių mokslų fakultetas

PAŽYMĖJIMAS

Nr. 24-0403-06

Tadas Švabauskas

2024 m. balandžio 3 d. dalyvavo
27-osios Lietuvos jaunųjų mokslininkų konferencijos
„Mokslas – Lietuvos ateitis“ teminėje konferencijoje
Informacinių technologijų sauga ir informacinės sistemos.
Sekcijoje „Informacinių technologijų sauga“ pristatė pranešimą
„Dinaminės kenksmingo programinio kodo elgsenos rinkimo pagerinimas panaudojant VMI“ („Improving the Collection of Dynamic Behavior of Malicious Code Using VMI“).

Mokslo komiteto pirmininkas prof. dr. Nikolaj Goranin

Organizacinio komiteto pirmininkas prof. dr. Diana Kalibatienė

Vilnius, 2024

1. Dinaminės kenksmingo programinio kodo elgsenos rinkimo metodų analizė

Šiame skyriuje bus apibrėžiama virtualizacijos sąvoka, aprašomi kenksmingo kodo analizės tipai. Išnagrinėjami dinaminės kenksmingo programinio kodo elgsenos rinkimo metodai bei jų trūkumai. Taip pat išanalizuojama VMI technika ir jos ypatumai.

1.1. Kenkėjiška programinė įranga

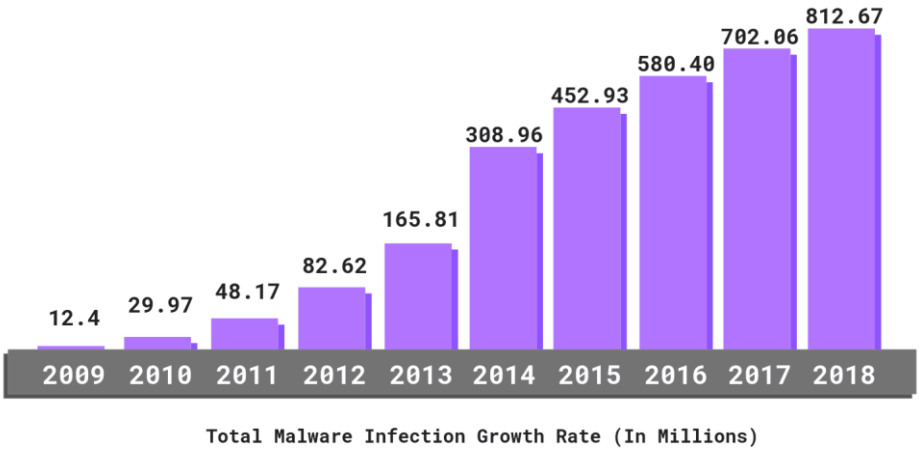
Prieš pasineriant į kenkėjiškų programų analizės ypatumus, geriau yra aptarti, kas gi yra kenkėjiškos programos. Taigi kenkėjiška programinė įranga yra programinė įranga, specialiai sukurta sutrikdyti kompiuterio, serverio, kliento ar kompiuterių tinklą, nutekinti asmeninę informaciją, gauti neteisėtą prieigą prie informacijos ar sistemų, neleisti vartotojams pasiekti informacijos ir pan. Galima išskirti keletą kenkėjiškų programų pavyzdžių:

- **„Virusai“** – kenkėjiškų programų pogrupis. „Virusas“ yra kenkėjiška programinė įranga, kurios tikslas – „apkrėsti“ sistemą bei reprodukuoti save į visas kitas įmanomas sistemas. Atsiųstas „virusą“ turintis failas savaime nepradeda veikti. Tai reiškia, kad reikalingas sistemos vartotojo įsikišimas (Alenezi et al., 2020). „Virusai“ yra skirti sutrikdyti sistemos gebėjimą veikti bei pažeisti duomenis.
- **Šnipinėjimo programos** – pagal Egele M., tai yra kenkėjiška programinė įranga, kuri bando tyliai stebėti sistemos vartotojų elgesį, fiksuoti jų naršymo internete įpročius arba pavogti slaptus duomenis, pvz., slaptažodžius (Egele et al., 2007). Užuot tiesiog sutrikdžiusi sistemos veiklą, šnipinėjimo programa nukreipiama į nieko neįtariančio sistemos vartotojo konfidencialios informacijos išgavimą, kuri atveria vagiams duris į įvairią vartotojo jautrią informaciją, pavyzdžiui, asmenines paskyras, banko sąskaitas ir t.t. Stebima informacija dažniausiai yra persiunčiama vagiams per internetinį tinklą.
- **„Kirminai“** – kitas kenkėjiškos programinės įrangos tipas, kuris greitai dauginasi ir plinta į bet kurią tinklo įrenginio sistemą. Skirtingai nuo „virusų“, „kirminai“ gali reprodukuoti save ir be sistemos vartotojo įsikišimo (Alenezi et al., 2020). „Kirminas“ užkrečia įrenginį per atsisiųstą failą arba tinklo ryšį bei dauginasi eksponentiniu greičiu. Kaip ir „virusai“, „kirminai“ gali labai sutrikdyti sistemos veikimą ir sugadinti ar pradanginti duomenis.
- **Išpirkos reikalaujančios programos** – kenkėjiška programinė įranga *(*angl. *Ransomware),* kuri sukurta atnešti pelną (Alenezi et al., 2020). Ji gauna prieigą prie neskelbtinos informacijos sistemoje, užšifruoja šią informaciją, kad vartotojas negalėtų jos pasiekti, ir tada reikalauja finansinės išmokos už duomenis, kurie paskiau

dešifruojami po išpirkos. Išpirkos reikalaujančios programos dažniausiai yra duomenų vagystės (išviliojimo) modelio dalis.

Galima teigti, kad kenkėjiškos programos taip pat gali būti platinamos per įvairius komunikacijos kanalus. „Phishing“ technika, kurią taiko kibernetiniai nusikaltėliai yra vienas iš pavyzdžių, kuomet vartotojas, nieko neįtardamas, vienu paspaudimu gali kompromituoti visą sistemą. Aleroud teigia, kad apytiksliai 95 % vartotojų nukenčia nuo kenkėjiškų programų padarinių, kuomet elektroniniuose laiškuose yra atidaromos nuorodos į kitus puslapius (Aleroud & Zhou, 2017). Duomenų išviliojimas (angl. terminas „phishing“ yra kilęs iš žodžių derinio – slaptažodžių žvejyba (angl. *password fishing*) – tai tokia sukčiavimo forma prieš organizacijas ar privačius asmenis, kai pasinaudojant nepageidaujamomis elektroninio pašto žinutėmis ar falsifikuotais internetiniais tinklalapiais siekiama išgauti prisijungimo prie informacinių sistemų slaptažodžius bei kitus konfidencialius duomenis (Nacionalinis kibernetinio saugumo centras, n.d.).

Remiantis populiariąją statistika, galima teigti, kad net 92 % kenkėjiškų programų yra platinamos elektroniniu paštu. Ypač 2018 metais 54 % kilstelėjo naujų kenkėjiškų programų, skirtų mobiliems įrenginiams, kiekis (Purplesec.us, 2021). Ši tendencija rodo, kad kuriamos naujos kenkėjiškos programos ir užkrečiamumo akiratis yra nuolatos plečiamas. 1.1 iliustracijoje pateikiama užkrečiamų įrenginių kiekis milijonais atitinkamais metais (2021 Cyber Security Statistics, 2021).



1.1 pav. KPK paveiktų sistemų augimas („2021 Cyber Security Statistics“, 2021)

Užpuolikų kuriamos kenkėjiškos programos gali būti:

- **Polimorfinės** – keičiamos identifikuojamos charakteristikos, pvz., failų pavadinimai ir tipai arba šifravimo raktai, kad kenkėjiška programa būtų neatpažįstama daugeliui aptikimo būdų.

- **Metamorfinės** – gali keisti savo kodą ir parašo šablonus su kiekviena iteracija. Metamorfinė kenkėjiška programa gali obfuskuoti visą savo kodą įvairiais būdais, pvz., valdymo srauto perkėlimu, lygiaverčių instrukcijų pakeitimu, kintamųjų pervadinimu ir kt. (Zhang & Reeves, 2007). Metamorfiniai virusai laikomi labiau pažengusiomis grėsmėmis nei įprastos kenkėjiškos programos ar net polimorfiniai virusai.

Kenkėjiškų programų įvairovė ir apimtis labai kenkia tradicinių apsaugos sistemų (antivirusinių) efektyvumui, kurios paprastai naudoja parašu pagrįstus metodus ir negali aptikti anksčiau nežinomų kenkėjiškų vykdomųjų failų. Kenkėjiškų programų šeimų variantai turi būdingus elgesio modelius, atspindinčius jų kilmę ir paskirtį. Statinės, dinaminės arba hibridinės analizės metu gautus elgesio modelius galima panaudoti, kad aptikti ir klasifikuoti nežinomas kenkėjiškas programas, naudojant mašininio mokymosi metodus (Gandotra et al., 2014).

1.2. Kenksmingo kodo ir programų analizių tipai

Aptikimas, identifikavimas ir išankstinė kenkėjiškų programų analizė yra labai svarbus aspektas, kad būtų užkirstas kelias kenkėjiškų programų plitimui. Praktikoje yra taikoma tiek statinė, tiek dinaminė analizė bei hibridinė analizė. Yra daug būtų analizuoti kenkėjiškas programas, jų kodą, bet nemažiau svarbu yra suprasti skirtumus tarp analizių tipų.

1.2.1. *Statinė analizė*

Šis metodas susijęs su „Portable“ vykdomųjų failų (PE failų) analize jų nepaleidžiant. Kenkėjiškų programų pavyzdžio kodas yra ištirtas pradine forma, siekiant nustatyti bet kokį kenkėjišką elgesį ar pavojaus požymius. Tai gali apimti paties kodo, taip pat bet kokių įterptųjų eilučių ar duomenų, kuriuose gali būti informacijos apie numatytą kenkėjiškos programos funkciją arba taikinius, analizę. Statinė analizė yra naudinga norint suprasti bendrą kenkėjiškų programų pavyzdžio struktūrą ir elgesį, taip pat nustatyti bet kokias unikalias savybes, kurios gali būti naudojamos ją aptikti arba blokuoti. Statinė analizė taip pat gali būti naudojama norint išgauti informaciją, kuri gali būti naudojama parašams generuoti, kad ateityje būtų aptikta kenkėjiška programa. Tačiau svarbu pažymėti, kad statinė analizė nesuteikia visos informacijos apie kenkėjiškų programų elgesį. Kai kurios kenkėjiškos programos naudoja tokius metodus kaip kodo supainiojimas, pakavimas ir anti-derinimas (angl. *anti-debugging*), kad išvengtų tokio tipo analizės.

Kenkėjiškos programos dažniausiai naudoja dvejetainį paketuotoją, pvz., „UPX“ ir „ASP Pack Shell“, kad būtų išvengta analizavimo. Prieš analizuojant PE failą, jį reikia išpakuoti ir dekompiliuoti. Norint dekompiliuoti „Windows“ vykdomąjį failą, galima naudoti išmontavimo įrankį, pvz., „IDA Pro“ ir „OlleyDbg“, kurie rodo surinkimo instrukcijas, pateikia informaciją apie

kenkėjišką programą ir ištraukia šabloną, kad būtų galima identifikuoti užpuoliką (Sihwail et al., 2018).

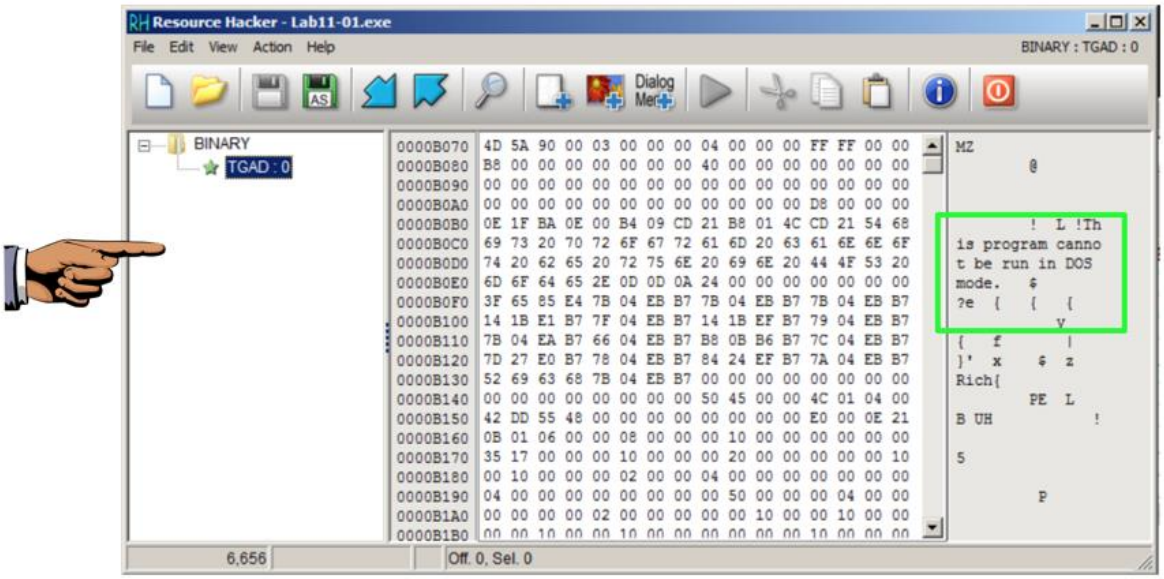
Taip pat gana patikimos alternatyvos bei papildomi įrankiai, tokie, kaip procesų analizių bei tinklo analizės įrankiai, šiems įrankiams yra:

- „PeStudio“ (išgauna statinius parametrus bei žymių anomalijas);
- „Process Hacker“ (stebi veikiančius procesus);
- „Process Monitor“ (kaupia lokalios sistemos veiksmus);
- „ProcDOT“ (filtruoja bei vizualizuoja „Process Monitor“ duomenis);
- „Wireshark“ (įrašinėja visus tinklo pokyčius);
- „Regshot“ (leidžia palyginti „Windows“ registrus, įžvelgi pakitimus) (Fox, 2021).

Aptikimo šabloną galima išgauti atliekant statinę analizę, pvz., „Windows“ API (angl. *Application Programming Interface*) sisteminius kreipinius, eilutės parašą, valdymo srauto grafiką („CFG“), operatyvinio kodo (operacijų kodų) dažnį ir baitų seką n-gramais.

Beveik visos programos naudoja „Windows“ API kreipinius, kad galėtų susisiekti su operacine sistema. Pavyzdžiui, „OpenFileW**“** metodas yra „Windows“ API, esantis „Kernel32.dll“ bibliotekoje, kuris sukuria naują failą arba atidaro esamą. API – tai taikomoji programinė sąsaja, kurią suteikia kompiuterinė sistema, biblioteka ar programa tam, kad programuotojas per kitą programą galėtų pasiekti jos funkcionalumą ar apsikeistų su ja duomenimis. Todėl API kreipiniai atskleidžia programų elgseną ir gali būti laikomi esminiu kenkėjiškų programų aptikimo ženklu.

Simbolių eilutės (angl. *strings*) (Strimaitis, n.d.) yra geras kenkėjiško kodo egzistavimo indikatorius. Jos atskleidžia užpuoliko ketinimus ir tikslus, nes jose dažnai yra svarbi semantinė informacija. Pavyzdžiui, dažnai pasitaikanti eilutė „Šios programos negalima paleisti DOS režimu“ nurodo galimai kenkėjišką failą, kai ji randama už tipinės PE antraštės (angl. *header*) pozicijos. Pavyzdys pateikiamas 1.2 iliustracijoje. Tai reiškia, kad greičiausiai tai yra „Trojos arklio“, skirto įdiegti tam tikrą kenkėjišką programą, bruožas, dar kitaip vadinamo – „dropper“. Kenkėjiškos programos „Trojan Dropper“ metu, vyksta kitų kenkėjiškų programų operacinėje sistemoje platinimas. Skirtingai nei „Trojan Downloader“, „Trojan Dropper“ programose kitos kenkėjiškos programos yra suspaustos arba užšifruotos jų pačių viduje (Gupta et al., 2018).



1.2 pav. Įtariamai kenkėjiškos programos bruožai („Proj 14: Malware Behavior“, n.d.)

Dvejetainio užmaskavimo metodai, kurie paverčia kenkėjiškų programų dvejetainius failus į savaime suspaustus ir unikalios struktūros dvejetainius failus, yra sukurti taip, kad atsispirtų atvirkštinei inžinerijai ir dėl to statinė analizė būtų labai brangi ir nepatikima. Be to, statinei analizei naudojant dvejetainius vykdomuosius failus (gautus kompiliuojant šaltinio kodą), prarandama tokia informacija kaip duomenų struktūrų ar kintamųjų dydis ir taip apsunkinama kenkėjiškų programų kodo analizė. Dėl besivystančių vengimo metodų, kuriuos naudoja kenkėjiškų programų kūrėjai, kad sutrukdytų statinei analizei, buvo sukurta dinaminė analizė (Moser et al., 2007), kurio nagrinėjo statinės analizės metodologijos trūkumus. Savo darbe jie pristatė kodo užmaskavimu pagrįstą schemą, atskleidžiančią faktą, kad vien statinės analizės nepakanka kenkėjiškoms programoms aptikti ar klasifikuoti. Be to, jie pasiūlė, kad dinaminė analizė būtų būtinas statinės analizės papildymas, nes ji yra mažiau pažeidžiama kodo užtemdymui (Gandotra et al., 2014).

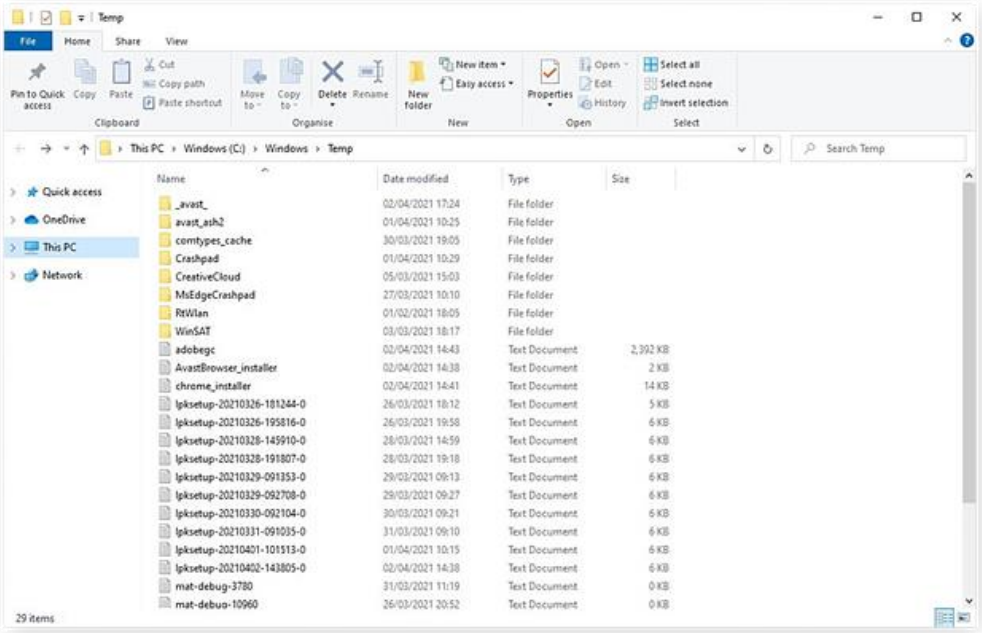
1.2.2. *Dinaminė analizė*

Ji taip pat vadinama elgesio analize. Šioje analizėje vykdomieji failai yra paleidžiami ir stebimi kontroliuojamoje aplinkoje, pvz., VM, emuliatoriuje ar simuliatoriuje, „smėlio dėžėje“. Užkrėstus failus reikia analizuoti nematomoje aplinkoje dėl paprastos priežasties – kai kurios kenkėjiškos programos palaikomos naudojant anti-virtualius įrenginius ir anti-emuliatorius. Aptikusios tokią aplinką kenkėjiškos programos elgiasi įprastai ir nerodo jokios kenkėjiškos veiklos.

Palyginti su statine analize, dinaminė analizė yra efektyvesnė, nes norint analizuoti, nereikia išardyti užkrėsto failo. Be to, dinaminė analizė gali aptikti žinomas ir nežinomas kenkėjiškas programas. Taipogi užmaskuota ir polimorfinė kenkėjiška programa negali išvengti dinaminio aptikimo. Tačiau dinaminė analizė reikalauja daug laiko ir daug išteklių (Sihwail et al., 2018).

Testuojamoje aplinkoje, virtualioje mašinoje, sudiegiami įvairūs analizavimo įrankiai, iš kurių dauguma atkeliauja iš statinės analizės, kurie yra paminėti ankstesniame skyriuje. Tačiau, kaipo buvo minėta, kiek pažangesnė kenkėjiška programa gali aptikti, o tiksliau – numatyti, kad ji yra testavimo aplinkoje. Tam reikalingas atitinkamas virtualios mašinos konfigūravimas ir paruošimas. Rekomenduojama izoliuotą testavimo aplinką paruoši kaip įmanoma geriau, pavyzdžiui:

- Suteikti logišką darbinių resursų kiekį, atitinkanti realios kompiuterinės įrangos atitikmenis, pavyzdžiui: ne mažiau, kaip 1-2 GB operacinės atminties (RAM), mažiausiai 2 procesoriaus branduolius, logišką pastoviosios atminties kiekį, kaip 128 GB. „Windows 10“ ir naujesnėms OS rekomenduojama skirti 4 GB ar daugiau operacinės atminties.
- Sudiegti OS atnaujinimus.
- Įrašyti papildomų programų, kurios imituotų kasdieninio naudojimo pagrindus, pavyzdžiui, „Google Chrome“ interneto naršyklę, „Notepad++“ teksto redaktorių, „Winamp“ muzikos grotuvą, „Torrent“ aplikaciją ir pnš.
- Kažkiek laiko simuliuoti įprastinį darbą su kompiuteriu, pavyzdžiui, naudotis įvairiomis programomis, kad „Windows“ užslėptose ir paprastam vartotojui galbūt nežinomose direktorijose, tokiose, kaip: „AppData“, būtų sukurti įvairūs laikini ir nelaikini failai su plėtiniais. Pavyzdys pateikiamas 1.3 pav.



1.3 pav. „Windows 10“ aplinkoje ‚cache‘ vadinami failai bei katalogai, sukurti po sistemos naudojimo, įdiegus įvairias programas ir jas panaudojus (Buxton, 2021)

- Pašalinti „VirtualBox Guest Additions“ virtualų apvaizdą, jeigu toks buvo pridėtas, kuris pagreitina ir optimizuoja virtualią aplinką. Kenkėjiškos programos gali aptikti ir tai ir riboti savo veikimą įtardamos, kad yra testavimo aplinkoje.
- Sukurti suklastotą internetinį tinklą. Tam yra skirtas patikimas įrankis „FakeNet“. Įrankis imituoja tinklą, kad kenkėjiška programa, sąveikaujanti su nuotoliniu serveriu, toliau veiktų ir analitikas galėtų stebėti kenkėjiškos programos tinklo veiklą saugioje aplinkoje.

Virtuali ir izoliuota aplinka, kurioje vykdomos kenkėjiškos programos, skiriasi nuo tikrosios, o kenkėjiškos programos gali veikti skirtingai, todėl ir jų „elgesys“ gali būti atitinkamai dirbtinis, suklastotas, o ne tikras. Be to, kartais kenkėjiškų programų elgesys suaktyvinamas tik tam tikromis sąlygomis (nustačius konkrečią sistemos datą ir laiką arba naudojant specifinę komandą) ir negali būti aptiktas virtualioje aplinkoje. Yra keletas internetinių automatinių įrankių bei „Windows“ operacinės sistemos sisteminių įrankių, iš kurių keletas (paryškinti) yra aptariami šiame darbe, skirtų dinaminei kenkėjiškų programų analizei, pavyzdžiui:

- „Norman Sandbox“,
- „CWSandbox“,
- „Anubis“,
- „TTAnalyzer“,
- „Ether“,
- „ThreatExpert“,
- **„Cuckoo sandbox“,**
- **„Drakvuf“, „Drakvuf Sandbox“,**
- „Windows Audit Logs“.

Gandotra ir kiti teigia, kad tokių įrankių sugeneruotos analizės ataskaitos suteikia išsamų supratimą apie kenkėjiškų programų elgesį ir vertingos informacijos apie jų atliekamus veiksmus. Analizės sistema turi turėti tinkamą kenkėjiškų programų, kurios vėliau naudojamos klasifikavimui pagal panašumo matą arba funkcijų vektorius, atvaizdą (Gandotra et al., 2014).

Tačiau daugybei naujų kenkėjiškų programų pavyzdžių, kasdien patenkančių į antivirusinių programų teikėjus, reikalingas automatinis metodas, kad būtų apribotas mėginių, kuriems reikalinga kruopšti žmogaus analizė, skaičius. Literatūroje automatizuotai kenkėjiškų programų analizei ir

klasifikavimui buvo naudojami keli dirbtinio intelekto metodai, ypač mašininiu mokymusi pagrįsti metodai (Gandotra et al., 2014).

Sihwail ir kiti teigia, kad dinaminei analizei gali būti naudojamos įvairios technikos, tokios kaip funkcijų kreipinių stebėjimas, funkcijų parametrų analizė, instrukcijų pėdsakai ir informacijos srauto sekimas. Peržiūrėjus tiriamus dokumentus, API ir sistemos kreipiniai dažniausiai naudojami kenkėjiškų programų dinaminei analizei, taip pat failų sistemai, „Windows“ registrui ir tinklo funkcijoms analizuoti (Sihwail et al., 2018).

1.2.3. *Hibridinė analizė*

Hibridinė analizė renka informaciją apie kenkėjiškas programas iš statinės ir dinaminės analizės. Naudodami hibridinę analizę, saugumo tyrinėtojai gauna tiek statinės, tiek dinaminės analizės pranašumus taip didinant galimybes teisingai aptikti kenkėjiškas programas. Abi analizės turi savų privalumų ir trūkumų. Statinė analizė yra pigi, greita ir saugesnė, palyginti su dinamine analize. Tačiau kenkėjiškos programos to išvengia naudodamos užmaskavimo būdus. Kita vertus, dinaminė analizė yra patikima ir gali įveikti užmaskavimo metodus. Be to, dinaminė analizė gali atpažinti kenkėjiškų programų variantus ir nežinomas kenkėjiškų programų šeimas. Tačiau tai reikalauja daug laiko ir resursų (Sihwail et al., 2018).

1.3. Virtualizacijos metodų iššūkiai

1.3.1. *„Smėlio dėžių“ veikimo keblumai*

„Smėlio dėžės“ paprastai kuriamos naudojant vieną iš dviejų tipų architektūrų: virtualizavimą arba emuliaciją. Virtualizacija pagrįstos „smėlio dėžės“ yra greičiau ir lengviau sukuriamos. Tačiau virtualizacija negali pilnai aptikti daugelių kenksmingų procesų, vykstančių kenksmingoje programoje, o tai labai apriboja virtualizacijos galimybes aptikti kenkėjiškas programas, kurios vengia būti aptiktos (Lastline, 2018).

Kita problema, su kuria susiduria virtualizacija pagrįstos „smėlio dėžės“, yra slaptumas. Visos „smėlio dėžės“ stengiasi būti neaptiktos kenkėjiškų programų, tačiau pažangi kenkėjiška programa gali aptikti hipervizoriaus technologiją, esančią visose virtualizacija pagrįstose „smėlio dėžėse“, ir paslėpti savo kenkėjišką elgesį, kad būtų išvengta aptikimo. Tai rimtas trūkumas (Lastline, 2018).

1.3.2. *Virtualizavimo aptikimo technikos, kurias taiko kenkėjiškos programos*

Yra trys technikos, kurias taiko pažangi kenkėjiška programa, kad išvengtų virtualizavimu pagrįstų „smėlio dėžių“:

- Virtualios mašinos artefaktų atradimas

- Virtualios mašinos dažnai savyje laiko specifinius artefaktus, kurie indikuoja į tai, kad tai yra „smėlio dėžės“ aplinka. Į artefaktus galima pažvelgti, kaip į „duonos trupinius“ – tai tokios specifinės detalės, kaip:
    - Virtualios mašinos įrankiai – pavyzdžiui, naudojant „VMware“ programinę įrangą, sistemoje gali užsilikti net virš 50 įvairių nuorodų, indikuojančių į „VMware“ programinę įrangą.
    - Konkretūs aparatinės įrangos parametrai, būdingi VM arba realiai fizinei aplinkai: juos galima rasti užklausant serijos numerius arba reikšmes, priskirtas procesoriams, pagrindinėms plokštėms ir t.t.
    - Registro raktuose gali būti daug reikšmių, kurias naudoja tik virtualios sistemos: „VMware“ gali saugoti daugiau nei 300 nuorodų į „VMware“.
    - Konkrečios sistemos atminties struktūros gali būti skirtingose VM aplinkos srityse, įskaitant pertraukimo deskriptorių lentelę (IDT) (angl. *Interrupt Descriptor Table*).
    - Virtualiojo tinklo adapterio MAC adreso arba BIOS serijos numerio identifikavimas: „VMware“ susieta su tam tikrais MAC adresais, prasidedančiais „00-05-69“, „00-0c-29“ ir kt.

- Vartotojo veiklos atradimas
    - Vartotojo veikla aplinkoje, įskaitant naršyklės žymes, istoriją, talpyklą ir pan., gali rodyti autentišką aplinką.
    - Kenkėjiška programa taip pat gali aptikti naudotojo veiklą pagal vartotojo sąveiką ir skaitmeninius parašus. Vartotojo aktyvumą galima nustatyti naudojant pelės paspaudimų dažnumą ir greitį – virtualioje sintetinėje aplinkoje jų bus tiek pat. Kiti įrodymai gali būti surinkti iš konkrečios vartotojo sąveikos su sistema prieš paleidžiant kenkėjišką programą. To pavyzdys gali būti laikas, per kurį vartotojas atlieka kenkėjiškos programos paleidimą du kart spustelėdamas programos ikonėlę.

- Virtualios aparatinės įrangos pirštų atspaudų atradimas
    - Kenkėjiška programa gali patikrinti sistemos komponentų temperatūrą (aušintuvo veikimo apsukų greitį), kad nustatytų, ar aplinka nėra virtuali. Sistemos procesoriaus aušintuvo patikrinimą galima atlikti naudojant tam

tikras WMI užklausas. Jei negaunami jokie rezultatai, tai gali reikšti, kad sistema greičiausiai yra virtuali.

Yra daug įvairių kenkėjiškų programų, kurios gali išvengti virtualizacijos „smėlio dėžių“, pavyzdžių. Toliau pateiktuose kenkėjiškų programų pavyzdžiuose naudojami aukščiau išvardyti metodai: „BadPatch“, „CHOPSTICK“, „CozyCar“, „Dyre“, „EvilBunny“, OopsIE, OSX_OCEANLOTUS.D, Remcos ir kt. (Belding, 2020).

1.4. Emuliacijos metodas – virtualizacijos alternatyva

Kaip alternatyvą virtualizacijai, kūrėjai gali naudoti emuliaciją kurdami „smėlio dėžę“ – arba emuliuodami operacinę sistemą (OS emuliacija), arba emuliuodami aparatinę įrangą (visa sistemos emuliacija). Sunkiau sukurti emuliacija pagrįstas „smėlio dėžes“, tačiau jos turi keletą pranašumų, palyginti su virtualizavimu pagrįstomis „smėlio dėžėmis“.

Aparatinės įrangos emuliacija yra metodas, kuriuo kompiuterinė sistema ar programa imituoja aparatinės įrangos veikimą. Tai reiškia, kad emuliatorius sukuria virtualią aparatinės įrangos aplinką, kuri veikia taip pat, kaip ir reali aparatinė įranga. Kenkėjiškų programų tyrimuose aparatinės įrangos emuliacija gali būti naudojama sukurti „smėlio dėžes“, kurios yra izoliuotos aplinkos, skirtos analizuoti kenkėjiškas programas. Tai leidžia analizuoti programos elgesį ir jos poveikį sistemai, be realios grėsmės sistemos saugumui. Aparatinės įrangos emuliacijos metodas yra labiau tikslesnis ir efektyvesnis aptikti kenkėjiškas programas, nes jis leidžia imituoti realias aplinkas, kuriose programos gali būti paleistos ir tai suteikia daugiau informacijos apie programos elgesį.

Kai kurios „smėlio dėžės“ emuliuoja operacinę sistemą. Idėja ta, kad emuliuojant operacinę sistemą, „smėlio dėžė“ geriau mato, ką daro kenkėjiška programa. Tačiau norint emuliuoti visą OS, reikia dubliuoti „Windows“ ir visas kitas įmonės OS. Tai, žinoma, neįmanoma. Taigi kūrėjai emuliuoja tik dalis OS – dažniausiai pasitaikančius ir taikomus sistemos kreipinius.

Problema ta, kad sukčiai perprato šį metodą, tad pažangi kenkėjiška programa sukelia retai naudojamus sisteminius kreipinius, kurių „smėlio dėžė“ nepalaiko. Dėl to kenkėjiška programa gali labai lengvai aptikti „smėlio dėžes“ ir išvengti jų, remiantis OS emuliacija.

Aparatinės įrangos emuliacija, kai emuliuojama visa sistema, įskaitant centrinį procesorių, atmintį ir įvesties / išvesties įrenginius (I/O), yra pažangesnis emuliavimo metodas, nes jis užtikrina didžiausią slaptumą ir didžiausią visų analizuojamų programų elgsenų analizę. Kadangi šis metodas nepalieka jokių artefaktų, jį aptikti daug sunkiau nei virtualizuojant ar emuliuojant OS. Visiška sistemos emuliacija taip pat leidžia „smėlio dėžei“ sąveikauti su kenkėjiška programa. Kai kenkėjiška

programa iškviečia operacinę sistemą, „smėlio dėžė“ gali pateikti skirtingus rezultatus ir įvertinti kenkėjiškos programos reakciją arba priversti vykdyti skirtingus kodo modulius. (Lastline, 2018).

Nepaisant iššūkių, virtualios aplinkos tyrimams skirti įrankiai yra nuolatos tobulinami siekiant kuo kruopščiau ištirti kenkėjišką programą ir bandyti apeiti jos ketinimus aptikti testavimo aplinką. Tam kuriami įvairūs įrankiai. Atsižvelgiant į tai, atitinkami įrankiai yra atrinkti ir įvertinti bei palyginti tolimesniuose skyriuose.

2. Virtual machine introspection (VMI) analizė

Terminą „introspekcija“ taikant virtualioms mašinoms sukūrė Garfinkelis ir Rosenblumas (Garfinkel & Rosenblum, 2003). Jie išrado metodą, kaip „apsaugoti saugos programą nuo kenkėjiškos programinės įrangos atakų“ ir pavadino ją VMI. Dabar VMI yra įprastas įvairių virtualių mašinų kriminalistikos ir analizės metodų terminas. VMI pagrįsti metodai plačiai naudojami saugos programoms, programinės įrangos derinimui ir sistemų valdymui.

VMI įrankiai gali būti virtualios mašinos viduje arba išorėje ir veikti sekdami įvykius (pertraukimus, įrašymą į atmintį ir pan.) arba siųsdami užklausas į virtualią mašiną. Virtualios mašinos monitorius paprastai teikia žemo lygio informaciją, pvz., neapdorotus atminties baitus. Šio žemo lygio rodinio pavertimas kažkuo reikšmingu vartotojui yra žinomas kaip semantinio tarpo problema (angl. *semantic gap problem*). Norint išspręsti šią problemą, reikia analizuoti ir suprasti stebimas sistemas.

Programos, veikiančios VM viduje, gali suteikti informacijos apie kitus procesus. Ši informacija gali būti siunčiama per tinklo sąsają arba kai kuriuos virtualius įrenginius, pvz., nuoseklųjį prievadą (angl. *serial port*). Vieni iš introspekcijos programų pavyzdžių yra „WinDbg“ arba „GDB“ serveriai, sąveikaujantys su nuotoliniu derintuvu. Šio metodo trūkumas yra tas, kad jam reikalinga veikianti operacinė sistema virtualioje mašinoje. Jei OS stringa arba dar nepakrauta – introspekcijos agentas negali veikti.

VMI įrankiai gali būti įdiegti virtualios mašinos monitoriuje arba kaip atskiros programos, fiksuojančios informaciją (pvz., atminties turinį) iš virtualios mašinos monitoriaus. Tada šie duomenys turi būti interpretuojami, kad būtų galima suprasti sistemoje vykstančius procesus. Viena iš populiarių tokio aiškinimo įrankių yra „Volatility“ karkasas (angl. *framework*), dažniausiai naudojamas kriminalistikos tyrimuose. Šioje sistemoje yra daugelio populiarių operacinių sistemų profiliai ir ji gali išgauti skirtingą informaciją, pvz., proceso medį arba branduolio objektų sąrašą.

VMI leidžia pasinaudoti technine įranga ir hipervizorių, kad galima būtų tikrinti „Guest“ (angl. *guest*) aplinką. Hipervizorius gali užprogramuoti aparatinę įrangą, kad gautų tam tikrus įvykius.

Tai naudinga virtualios mašinos savistabai, nes leidžia užfiksuoti konkrečius veiksmus, kuriuos „Guest“ VM gali atlikti, ir patikrinti svečio būseną būtent tuo momentu. Tai veda prie kito pranašumo: hipervizorius puikiai mato visą svečio būseną. Tai apima procesorius, atmintį ir visus įrenginius (pvz., tinklo plokštes, standžiuosius diskus ir kt.). Galiausiai, VMI komponentai yra visiškai izoliuoti nuo tikrinamos aplinkos. Tai apsunkina tikrinančių komponentų aptikimą ar atakavimą ir sumažina stebėtojo poveikį svečio aplinkai (BedRock, 2020).

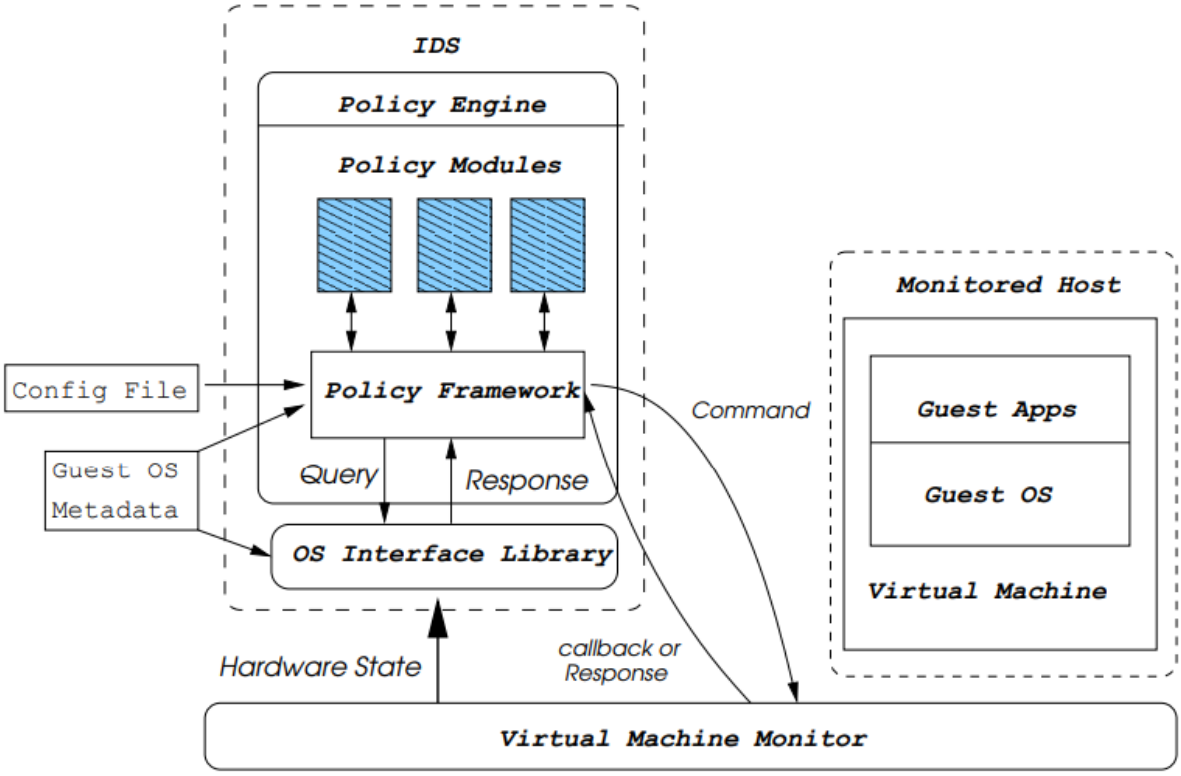
Pritaikius VMI galima nustatyti lūžio taškus svečio viduje ir perduoti juos hipervizoriui kiekvieną kartą, kai įvykdomas lūžio taškas. Pasitelkus virtualios mašinos savianalize, galima nustatyti tokį lūžio tašką, kad „Guest“ VM negalėtų matyti lūžio taško atmintyje arba aptikti, kad prie proceso yra prijungtas derinimo įrankis. Kartais, kaip buvo minėta ankstesniame skyriuje, programa gali pakeisti savo elgesį, kai nustato, kad ji yra tikrinama. Ši antianalizės forma yra paplitusi kenkėjiškose programose arba programinėje įrangoje, kuri bando apsaugoti savo intelektinę nuosavybę (BedRock, 2020).

VMI leidžia tiesiogiai apžiūrėti virtualią aparatūrą, suteikiant unikalų sistemos vaizdą, kuris neįmanomas jokiu kitu būdu (t. y. visa fizinė atmintis ir visi procesoriaus registrai visada pasiekiami). Pavyzdžiui, galima apsaugoti atmintį nuo modifikacijų, apžiūrėti kiekvieną procesą prieš jam įvykstant arba sugeneruoti visą valdymo srauto grafiką ir visa tai galima padaryti visiškai atskyrus nuo svečio ir visų komponentų, kurie gali trukdyti tyrimui (BedRock, 2020).

Virtualios mašinos monitoriaus (VMM) ir hipervizoriaus komunikacijai VMM architektūroje vyrauja 3 esminiai sąveikavimo komandų tipai:

- Tikrinimo komandos – naudojamos tiesiogiai ištirti VM būseną, pvz., atminties ir registro turinį bei įvesties / išvesties įrenginių parametrus.
- Monitoriaus komandos – naudojamos aptikti, kada įvyksta tam tikri mašinos įvykiai, ir prašyti pranešimo per įvykių pristatymo mechanizmą. Pavyzdžiui, VMI gali gauti pranešimą, kai pasikeičia tam tikras atminties diapazonas, privilegijuotas registras arba pasikeičia įrenginio būsena (pvz., „Ethernet“ sąsajos adreso pasikeitimas).
- Administracinės komandos – leidžia VMI įsibrovimų aptikimo sistemai (IDS) valdyti VM vykdymą. Ši sąsaja leidžia VMI IDS sustabdyti VM vykdymą, atnaujinti sustabdytą VM, patikrinti VM ir iš naujo paleisti VM. Šios komandos pirmiausia naudingos norint paleisti sistemą ir automatizuoti atsaką į kompromisą. VMI IDS suteikiama tik stebimos VM administracinė kontrolė.

Platesnis vaizdas pateikiamas VMI architektūros schemos 2.1 pav.:

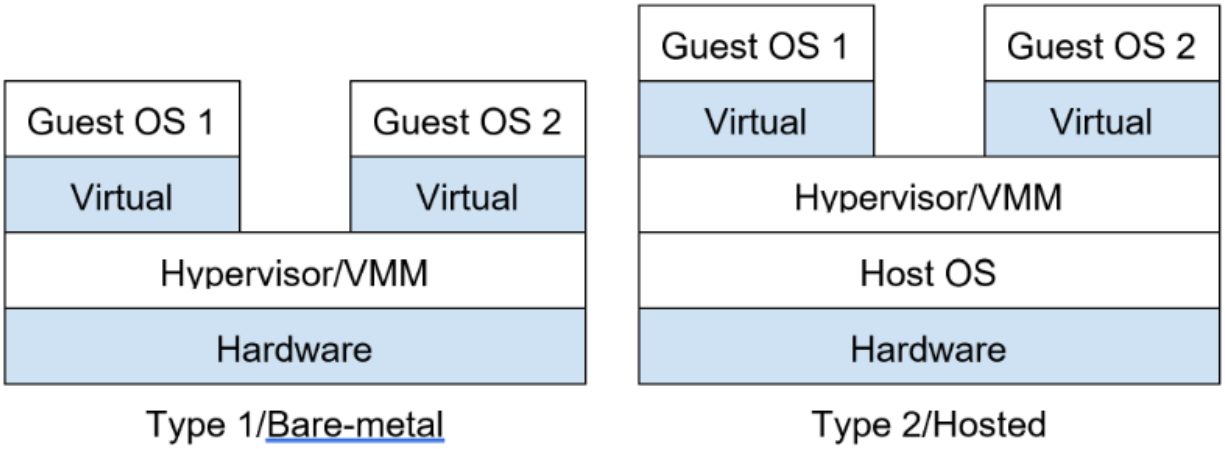


2.1 pav. VMI architektūros schema (Garfinkel & Rosenblum, 2003)

2.1.2. *Hipervizorių tipai*

Yra du pagrindiniai hipervizorių tipai: (2.2 pav)

- 1 tipas „bare-metal“,
- 2 tipas „hosted“.



2.2 pav. Hipervizorių tipai

1 tipo hipervizoriai, taip pat žinomi kaip vietiniai arba pliko metalo hipervizoriai, veikia tiesiogiai pagrindinio kompiuterio aparatinėje įrangoje. 1 tipo hipervizorių pavyzdžiai yra „VMware ESXi“, „Microsoft“ „Hyper-V“ ir „Citrix XenServer“. Šie hipervizoriai turi tiesioginę prieigą prie

pagrindinio kompiuterio aparatinės įrangos išteklių, o tai leidžia užtikrinti aukštą našumo ir saugumo lygį. Jie taip pat gali visiškai valdyti virtualizuotą aplinką, o tai leidžia teikti pažangias funkcijas, tokias kaip tiesioginis perkėlimas ir atminties dubliavimas.

2 tipo hipervizoriai, taip pat žinomi kaip „hosted“ hipervizoriai, veikia pagrindinės operacinės sistemos viršuje. 2 tipo hipervizorių pavyzdžiai yra „Oracle VirtualBox“, „VMware Workstation“ ir „Parallels Desktop“. Šie hipervizoriai neturi tiesioginės prieigos prie pagrindinio kompiuterio aparatinės įrangos išteklių ir pasikliauja pagrindine operacine sistema, kad suteiktų prieigą prie aparatinės įrangos išteklių. Dėl to gali sumažėti našumo ir saugumo lygis, palyginti su 1 tipo hipervizoriais. Tačiau juos paprastai lengviau naudoti ir juos galima paleisti su įvairia aparatūra.

VMI kontekste labiau tinka 1 tipo hipervizoriai, kurie turi tiesioginę prieigą prie aparatinės įrangos ir gali užtikrinti aukštą virtualizuotos aplinkos našumo, saugumo ir kontrolės lygį. Tai leidžia jiems teikti pažangesnes funkcijas, pvz., atminties patikrinimą, kuris yra labai svarbus kenkėjiškų programų analizei ir atsakui į incidentus.

2 tipo hipervizoriai gali būti naudojami VMI, tačiau jiems gali trūkti kai kurių pažangių funkcijų, kurias teikia 1 tipo hipervizoriai, taip pat gali būti žemesnis našumo ir saugumo lygis.

2.1.3. „LibVMI“

„LibVMI“ (angl. *Library for Virtual Machine Introspection*) yra atviro kodo introspekcijos biblioteka, skirta atminties skaitymui ir rašymui iš ir į VM. Patogumui „LibVMI“ taip pat teikia funkcijas, skirtas pasiekti procesoriaus registrus, pristabdyti ir atšaukti VM, spausdinti dvejetainius duomenis ir kt. „LibVMI“ sukurta dirbti keliose virtualizacijos platformose. „LibVMI“ šiuo metu palaiko VM, veikiančias per „Xen“ arba „KVM“ („Qemu“). „LibVMI“ taip pat palaiko fizinės atminties momentinių nuotraukų skaitymą, kai jie išsaugomi kaip failas („Introduction to LibVMI“ n.d.).

„LibVMI“ suteikia daugybę funkcijų, dėl kurių ji naudinga atliekant kenkėjiškų programų analizę, reaguojant į incidentus ir atliekant kitas su sauga susijusias užduotis. Kai kurios pagrindinės „LibVMI“ funkcijos:

- Atminties introspekcija: „LibVMI“ leidžia ištirti virtualios mašinos atmintį, kuri gali būti naudojama kenkėjiškų programų identifikavimui ir analizei.
- Įvykių stebėjimas: „LibVMI“ gali perimti ir tvarkyti įvykius, tokius kaip proceso kūrimas, sistemos kreipiniai ir tinklo veikla. Tai gali būti naudojama kenkėjiškų programų elgsenai aptikti ir analizuoti.

- Operacinės sistemos nepriklausomybė: „LibVMI“ palaiko ir „Windows“, ir „Linux“ operacines sistemas, todėl ją naudinga analizuoti platų kenkėjiškų programų asortimentą.
- Hipervizoriaus nepriklausomybė: „LibVMI“ palaiko ir 1, ir 2 tipo hipervizorius, todėl tai yra naudinga analizuojant kenkėjišką programą įvairiose virtualizuotose aplinkose.

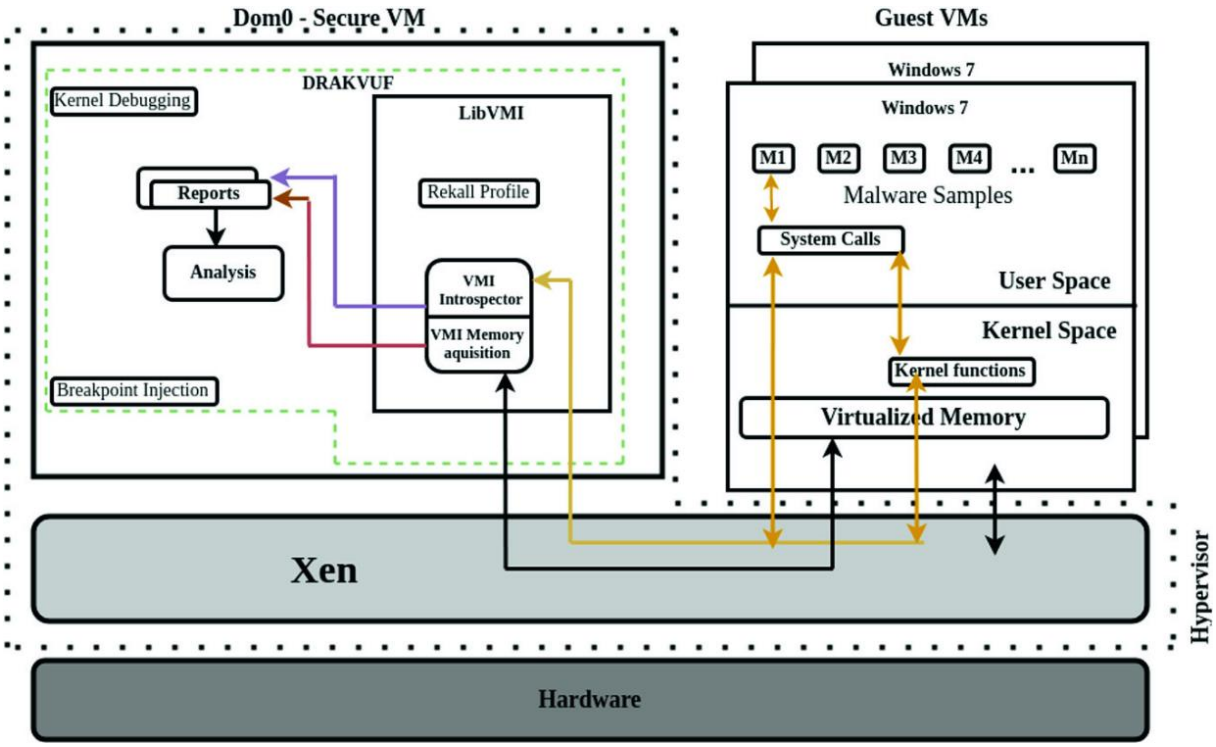
„LibVMI“ gali būti naudojama kartu su kitais įrankiais, tokiais kaip „Drakvuf“, „Rekall“ ir „Volatility“ kenkėjiškų programų analizei, reagavimui į incidentus ir kitoms su saugumu susijusioms užduotims atlikti.

2.2. „Drakvuf-VMI“

„Drakvuf“ yra virtualizacija pagrįsta juodosios dėžės dvejetainės analizės sistema be agentų, kurią sukūrė Lengyel. „Drakvuf“ leidžia nuodugniai sekti dvejetainių failų (įskaitant operacines sistemas) vykdymą, neįdiegiant jokios specialios programinės įrangos analizei naudojamoje testavimui virtualioje mašinoje. Tai VMI pagrįsta technika, veikianti branduolio derinimo (angl. *kernel debugging*) principu. „Drakvuf“ turi galimybę užfiksuoti visas branduolio funkcijas naudojant #BP injekcijos techniką. „Drakvuf“ architektūros schema parodyta 2.3 pav.

Sistema sukurta naudojant „Xen“ virtualizacijos platformą ir naudoja ankstesniame skyriuje analizuotą „LibVMI“ API biblioteką. Kaip minėta, „LibVMI“ yra sukurta iš „XenAccess“ bibliotekos. VMI introspekcijos komponentas yra atsakingas už sistemos kreipinių sekimą, branduolio lygmens funkcijas, kurias iššaukia kenkėjiška programa, taip pat tokias failų operacijas, kaip failų kūrimą ir ištrynimą, kuriuos atlieka kenkėjiška programa.

Sukurti ir ištrinti laikinieji failai atsekami atsargiai išskaidžius atminties sąstingio iškrovą (angl. *memory dump*). Kenkėjiškos programinės įrangos vykdymo metu esančią atminties iškrovą surenka VMI atminties gavimo modulis. Norint sėkmingai atlikti introspekcijos operaciją, būtina iš anksto žinoti apie testuojamos aplinkos OS vidinę duomenų struktūrą. „Windows“ operacinėje sistemoje tai galima padaryti sukuriant „Rekall“ profilį naudojant „Microsoft“ pateiktą „Windows“ informaciją (Melvin & Kathrine, 2021).



2.3 pav. „Drakvuf-VMI“ architektūros schema (Melvin & Kathrine, 2021)

Taigi, galima teigti, kad „Drakvuf“ yra puiki platforma slaptai kenkėjiškų programų analizei, nes jos pėdsakas yra beveik nepastebimas iš kenkėjiškų programų perspektyvos. Nors „Drakvuf“ daugiausia buvo sukurta turint omenyje kenkėjiškų programų analizę, ji tikrai neapsiriboja šia užduotimi, nes ji gali būti naudojama savavališkų dvejetainių failų vykdymui stebėti.

Sekančiame skyriuje apžvelgsime „Cuckoo Sandbox“ įrankį, kuris nėra paremtas VMI technika. Tai padės geriau suprasti analizės metodus, naudojant įvairius agentus „Guest“ virtualios mašinos aplinkoje.

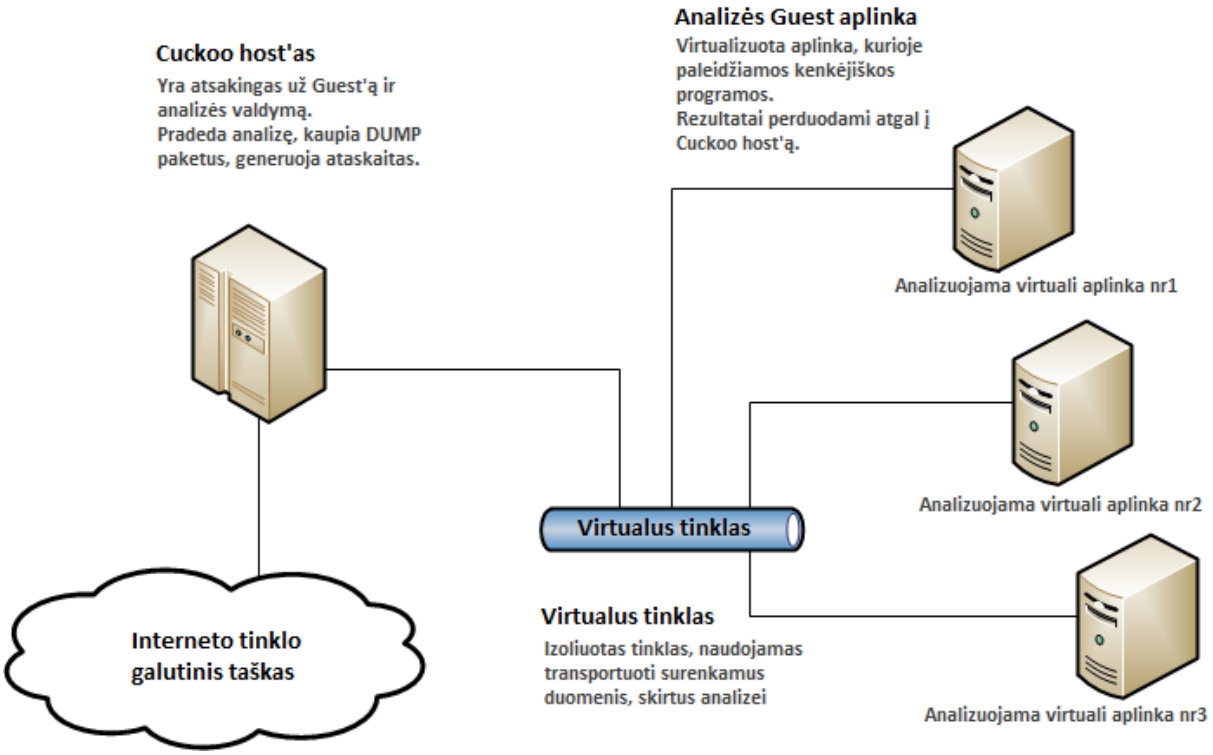
2.3. „Cuckoo Sandbox“

„Cuckoo Sandbox“ yra pirmaujanti atvirojo kodo itin modulinė automatizuota kenkėjiškų programų analizės sistema (Stichting Cuckoo Foundation, n.d.). Ši programinė įranga yra nemokama, kuri automatizuoja bet kokio kenkėjiško failo analizę sistemoje „Windows“, „MacOS“, „Linux“ ir „Android“ operacinėse sistemose. Pagal numatytuosius nustatymus ji gali:

- Analizuoti daugybę įvairių kenkėjiškų failų (vykdomųjų failų, biuro dokumentų, „pdf“ failų, el. laiškų ir t. t.), taip pat kenkėjiškas svetaines naudojant „Windows“, „MacOS“, „Linux“ ir „Android“ virtualizuotą aplinką.
- Stebėti API kreipinius ir bendrą failo veikimą ir išskirti tai į aukšto lygio informaciją ir parašus, suprantamus visiems.

- Kaupti atminties iškrovos paketus ir analizuoti „SSL“ ir „TLS“ užšifruotą tinklo srautą.
- Atlikti išplėstinę užkrėstos virtualizuotos sistemos atminties analizę naudodami „Volatility“, taip pat proceso atminties detalumo analizę naudojant „YARA“.

Šis paveikslėlis pav. 2.4, paaiškins pagrindinę „Cuckoo“ architektūrą:



2.4 pav. Pagrindinė „Cuckoo“ architektūra (Stichting Cuckoo Foundation, n.d.)

Architektūrą sudaro tokie komponentai: „Host“ mašina, analizės tvarkyklė, virtualus tinklas, analizės „Guest“ virtualios mašinos (VM1, VM2, VM3) ir agento modulis. Vartotojas sąveikauja su „Cuckoo“ iš pagrindinio kompiuterio per komandinės eilutės sąsają arba per žiniatinklio sąsają. Analizės vadovas yra atsakingas už kenkėjiškų programų pavyzdžių pateikimą ir ataskaitų rinkimą baigus kenkėjiškų programų analizę. Virtualizavimo platforma yra atsakinga už virtualių mašinų, skirtų kenkėjiškų programų vykdymui, sukūrimą. „Cuckoo“ turi galimybę pateikti užduotis kelioms virtualioms mašinoms, kad būtų vykdomi kenkėjiškų programų pavyzdžiai. „Cuckoo“ „Host“ mašina sąveikauja su analizės virtualiomis mašinomis per virtualų tinklą. Virtualioje mašinoje įdiegta testavimo aplinkos OS suteikia realią aplinką kenkėjiškų programų pavyzdžiams vykdyti. Prieš pradedant analizę yra padaromas momentinis VM atvaizdas (angl. *snapshot*) ir jis vėliau yra naudojamas, kad grąžinti VM į pradinę būseną, t. y. prieš paleidžiant kenkėjišką programą. Tai užtikrina, kad kitas kenkėjiškos programos vykdymas bus atliktas tuščioje, neužkrėstoje,

nesugadintoje OS testavimo aplinkoje. VM „Python“ agento modulis yra įdiegtas į testavimo OS, sukonfigūruotas taip, kad būtų paleistas automatiškai. Analizės tvarkyklė sąveikauja su testavimo OS per šį „Python“ agentą. „Python“ agentas yra atsakingas už kenkėjiškos programos gavimą, jos vykdymą ir analizės ataskaitos siuntimą analizės tvarkytojui (Melvin & Kathrine, 2021).

Sekančiame skyriuje palyginsime anksčiau minėtą pažangesnės technologijos VMI paremtą „Drakvuf“ ir „tracidinės“ dinaminės analizės sistemą – „Cuckoo“ bei įžvelgsime „Drakvuf“ pranašumą taikant VMI.

2.4. „Cuckoo“ ir Drakvuf-VMI palyginimas

Melvin ir Kathrine (2021) mokslinėje literatūroje yra pateikę „Cuckoo“ „smėlio dėžės“ bei „Drakvuf“ VMI technika grįstų kenkėjiškų programų analizių įrankių palyginimas. Jų atliktame tyrime pateikiami kriterijų palyginimai tarp skirtingų metodų. Taip pat atliktų testavimų rezultatai, kuriuos sudaro:

- bendrinė eksperimentų laiko analizė, kuomet leidžiamos kenkėjiškos programos ir yra surenkami sistemos kreipiniai;
- pralaidumo analizė, kuomet įvertinamas kenkėjiškų programų bei laiko sąmatos santykis.

Vertinant pagal kriterijus, pagrindines įžvalgas sudaro tai, kad „Cuckoo“ turi esminį trūkumą – nėra slaptumo. Kenkėjiška programa, tai numatydama, bando nuslėpti savo kenkėjišką funkcionalumą. Iš kitos pusės, „Drakvuf“ nenaudoja jokio agento, taip užtikrindama slaptumą, savęs maskavimą, testuojamoje aplinkoje. Tad kenkėjiška programa elgsis natūraliai. Kitas esminis skirtumas – „Drakvuf“ gebėjimas atsekti visas, tiek vartotojo aplinkoje, tiek branduolio lygmenyje užklausas, sistemos kreipinius bei funkcijas, kuriomis naudojasi kenkėjiška programa. „Cuckoo“ negali atsekti sistemos kreipinių branduolio lygmenyje, tačiau gali vartotojo aplinkoje. „Drakvuf“ taip pat turi galimybę atsekti branduolio „rootkit“-ų funkcionavimą. Abu metodai turi galimybę išgauti bei perduoti testavimo metu gautus duomenis tiek į kitas virtualias mašinas, tiek į šeimininko sistemą. Taip pat abu metodai gali būti plečiami. Abi sistemos yra atviro kodo ir yra daugi modulinės sistemos, kurios gali būti integruojamos su kitomis sistemomis. Kitokie atviro kodo komponentai gali būti įtraukiami į šias sistemas. Abu metodai palaiko ne mažą kiekį operacinių ir jų versijų kiekį. Šie esminiai palyginimo kriterijai toliau pateikiami 2.1 lentelėje:

2.1 lentelė. „Cuckoo“ ir „Drakvuf“ palyginimas (Melvin & Kathrine, 2021)

#	Kriterijai/parametrai	„Cuckoo Sandbox“ metodas	„Drakvuf-VMI“ metodas
1	Slaptumas	-	+

		Kadangi yra naudojamas specifinis komunikacijos agentas testuojamoje aplinkoje, kenkėjiška programa gali tai aptikti ir nuslėpti savo kenkėjiškas funkcijas.	Kadangi nėra naudojamas agentas, kenkėjiška programa negali nieko įtarti, vadinasi ji veikia natūraliai.
2	Sistemos kreipiniai	+ Gali atsekti visus sistemos kreipinius, iniciuotus kenkėjiškos programos.	+ Gali atsekti visus sistemos kreipinius, iniciuotus kenkėjiškos programos.
3	Branduolio funkcijos	- Negali atsekti branduolio lygmens iškviečiamų funkcijų.	+ Gali atsekti visas branduolio lygmenyje iškviečiamas funkcijas.
4	Automatizavimas	+ Palaiko automatinį kenkėjiškų programų pavyzdžių pateikimą. Nereikalingas bet koks papildomas rankinis įsikišimas.	+ Taip pat palaiko automatinį kenkėjiškų programų pavyzdžių pateikimą. Nereikalingas bet koks papildomas rankinis įsikišimas.
5	Išplėstinumas	+ Gali būti išplečiama padidinus virtualių mašinų aplinkų kiekį, naudojama analizei. Tačiau tai priklauso nuo „Host“ mašinos fizinės atminties dydžio, skirtos kaupti analizės duomenims.	+ Gali būti išplečiama padidinus virtualių mašinų aplinkų kiekį, naudojama analizei. Tačiau tai priklauso nuo „Host“ mašinos fizinės atminties dydžio, skirtos kaupti analizės duomenims.
6	OS palaikymas	+ „Windows“ XP/7/8/10, „Linux“ 2.6 ir naujesnė.	+ „Windows“ XP/7/8/10, „Linux“ 2.6 ir naujesnė.
7	Izoliacija	+ Aprūpina izoliuotą aplinką kenkėjiškų programų analizei.	+ Aprūpina izoliuotą aplinką kenkėjiškų programų analizei.
8	Atminties momentinis fiksavimas	+ Turi galimybę ištraukti testuojamos aplinkos atmintį (angl. *dump memory*).	+ Turi galimybę ištraukti testuojamos aplinkos atmintį (*angl. dump memory*).

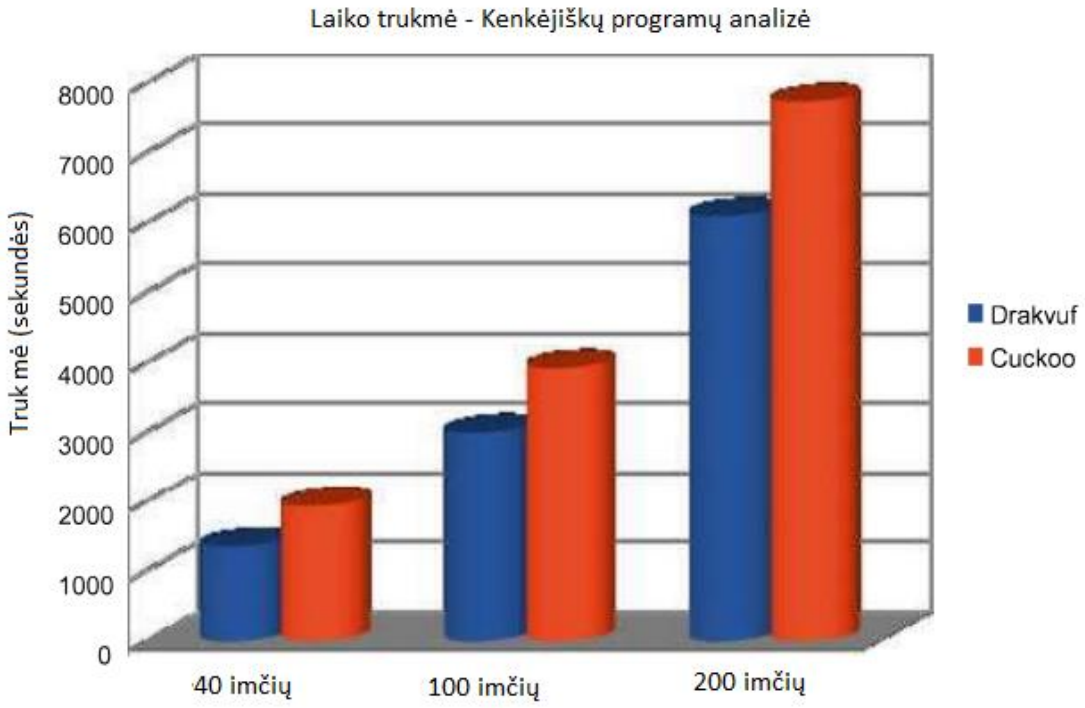
Tyrimo trukmės eksperimentas atliekamas tokia tvarka: kenkėjiškų programų pavyzdžiai yra perkelti į dvi testavimo aplinkas – „Host 1“ ir „Host 2“. Abiejose aplinkose buvo įjungtas atminties išmetimas (angl. *memory dumping*) ir nustatytas 60 sekundžių analizės laiko limitas.

„Host 1“ aplinkoje paleidžiama „shell“ rašmena (angl. *script*), kuri sukuria dvi VM. Bendras kenkėjiškų programų skaičius padalinimas į dvi puses ir tos pusės paruošiamos perkelti į abi VM. Fiksuojamas pirminis laikas prieš pradedant analizę. Kenkėjiškos programos vykdymą VM viduje inicijuoja „Drakvuf“ injekcijos technika. Dėl šios priežasties VM nereikia jokio svečio modulio. Baigus visų kenkėjiškų programų analizę, fiksuojamas pabaigos laikas. Atėmus pabaigos laiką iš pradžios, gaunama visa apdorojimo trukmė.

„Host 2“ kenkėjiškos programos perkeliamos per „Cuckoo“ analizės tvarkyklę į vieną iš galimų VM. VM viduje esantis agento modulis inicijuoja kenkėjiškos programos vykdymą ir renka visus elgsenos duomenis. Galiausiai surinktus duomenis siunčia atgal „Cuckoo“ valdytojui. Ši

operacija yra pakartojama su visais kenkėjiškų programų pavyzdžiais. Užbaigus visų kenkėjiškų programų rinkinio analizę, „Cuckoo“ žurnalo failas yra apdorojamas, siekiant gauti analizės pradžios ir pabaigos laiką. Lygiai taip pat atėmus pabaigos laiką iš pradžios, gaunama visa apdorojimo trukmė.

Eksperimentai buvo atliekami su 40, 100 ir 200 imčių. Analizės rezultatai pateikiami 2.5 paveikslėlio grafike. O detalūs duomenys aprašomi 2.2 lentelėje.



2.5 pav. Kenkėjiškų programų skirtingų metodų analizių trukmės (Melvin & Kathrine, 2021)

2.2 lentelė. Kenkėjiškų programų skirtingų metodų analizių trukmės (Melvin & Kathrine, 2021)

	40 imčių	**100 imčių**	**200 imčių**
Drakvuf	22 minutės 34 sekundės, arba –1354 sekundės	50 minučių, arba – 3000 sekundės	1 valanda 41 minutės 39 sekundės, arba – 6099 sekundės
Cuckoo	32 minutės 16 sekundžių, arba – 1936 sekundės	1 valanda 5 minutės 23 sekundės, arba – 3923 sekundės	2 valandos 8 minutės 53 sekundės, arba – 7733 sekundės

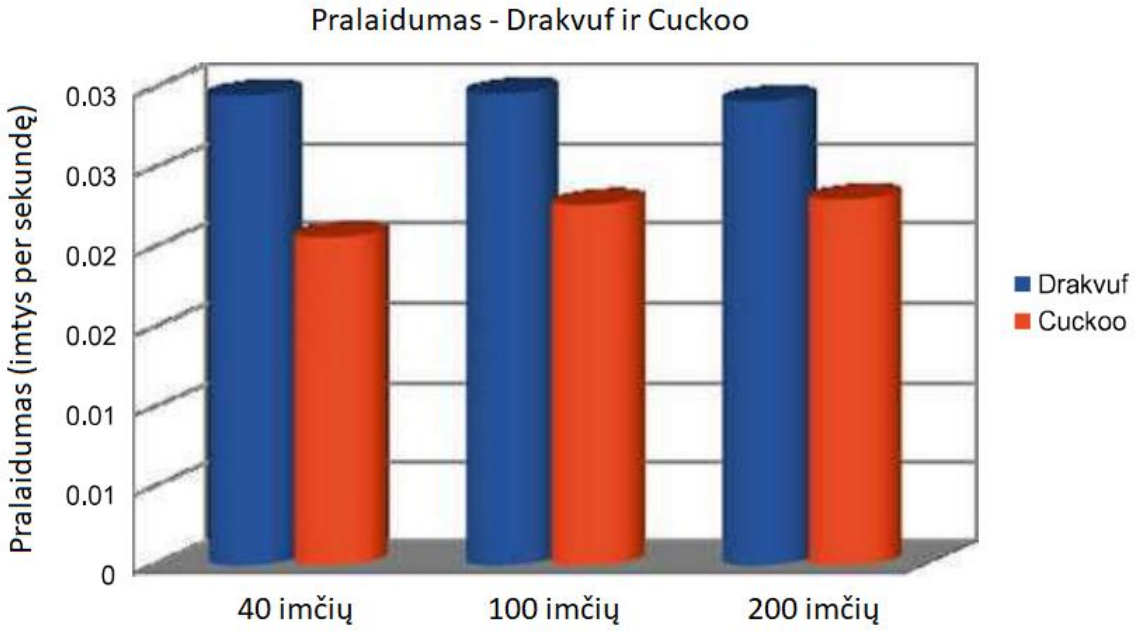
Pralaidumas yra išanalizuotų kenkėjiškų programų kiekis per sekundę. Jis buvo apskaičiuotas pagal formulę:

Pralaidumas = galutinis skaičius sėkmingai išanalizuotų imčių / trukmė

Gauti rezultatai pateikiami sekančioje 2.3 lentelėje bei 2.6 paveikslėlio grafike:

2.3 lentelė. Kenkėjiškų programų pralaidumas (Melvin & Kathrine, 2021)

	40 imčių	100 imčių	200 imčių
„Drakvuf“	0.029542097	0.029666667	0.029185112
„Cuckoo“	0.020661157	0.022686719	0.023018234



2.6 pav. Kenkėjiškų programų pralaidumas (Melvin & Kathrine, 2021)

Laiko analizės eksperimento metu atminties išmetimas buvo įjungtas atitinkamai *Drakvuf* ir „Cuckoo“, kad būtų galima sužinoti disko panaudojimą analizės metu. Abi sistemos sukūrė įvairių ataskaitų failų.

„Host 1“ kekėjiškų protramų analizės metu, „Drakvuf“ sukuria žurnalo failą „*Drakvuf.log*“ „*csv“* formatu ir failus su plėtiniais, tokiais kaip metaduomenys, „*mm“* ir „*dmp“*. Metaduomenų failuose yra sukurtas arba pakeistas failo pavadinimas, failo dydis, failo vėliavėlės (angl. *flags*), „PID“ (proceso identifikacinis numeris), PPID (tėvinis identifikacinis numeris) ir proceso pavadinimas. „.dmp“ failai yra atminties išklotinės, sukurtos vykdant kenkėjišką programą. Be šių failų, buvo naudojamas „tcpdump“, kad užfiksuoti visus paketus, gautus iš analizės VM. „Tcpdump“ išvestis saugoma kaip tcpdump.log.

„Host 2“ kenkėjiškų programų analizės metu, „Cuckoo“ taip pat sukuria įvairius failus. *Cuckoo.log* faile yra išsami informacija apie dabartinę užduotį. Pagrindinis „Cuckoo“ failas yra „report.json“. „Report.json“ faile struktūrizuotai saugomi visi konkrečios kenkėjiškos programos aspektai. Pavyzdžiui, kenkėjiškos programos naudojama importuota funkcija ir jos adresas, importuoti „DLL“ bibliotekų failai, naudojami registro raktai, naudojami sistemos kreipiniai ir t.t. Be visų failų buvo sukurtas ir virtualios atminties išmetimas – „memory.dmp“.

Sistemos naudojamos vietos kietajame diske naudojimas įvertinamas naudojant „du“ („disk usage“) komandą, o jo naudojimo procentas apskaičiuojamas ir pateikiamas 2.4 lentelėje:

2.4 lentelė. Vietos diske naudojimas su atminties išmetimu ir be jo (Melvin & Kathrine, 2021)

	100 imčių (atminties išmetimas naudojamas)	**100 imčių (atminties išmetimas NEnaudojamas)**
„Drakvuf“	5959050550 bitai (5.959051 %)	5958980465 bitai (5.95898 %)
„Cuckoo“	110151000000 bitai (11.0151 %)	155347776 bitai (0.0155348 %)

Iš 3.4 lentelės nustatyta, kad „Cuckoo“ sunaudoja daugiau vietos, palyginti su „Drakvuf“, kai įjungtas atminties išmetimas. Tačiau išjungus atminties išmetimą, „Cuckoo“ sunaudoja labai mažiau vietos, palyginti su „Drakvuf“.

3. Sisteminių kreipinių duomenų surinkimo metodo tyrimas

Šiame skyriuje apžvelgiamas „Windows“ sisteminių kreipinių duomenų surinkimo metodas bei AWSCTD duomenų rinkinio sudarymas mašininiam mokymui.

3.1. DrStrace su ProxMox įgyvendintas metodas

Čeponis ir Goranin (2018) pristatytoje mokslinėje literatūroje, „*Towards a Robust Method of Dataset Generation of Malicious Activity for Anomaly-Based HIDS Training and Presentation of AWSCTD Dataset“,* pateikia jų sukurtą KPK iškviestų sistemos kreipinių rinkimo metodą panaudojant „drStrace“ įrankį, „ProxMox“ hipervizorių bei kitus papildomus apdorojimo įrankius, skirtus darbui su duomenimis.

„ProxMox“ virtualizacijos aplinka (VE) yra atviro kodo platforma su integruota žiniatinklio sąsaja, suteikianti galimybes administruoti virtualias mašinas. „ProxMox“ įrankyje veikia „QEMU“ – bendras atvirojo kodo mašinos emuliatorius ir virtualizatorius, pagrįstas „Debian“ operacine sistema (Čeponis & Goranin, 2018).

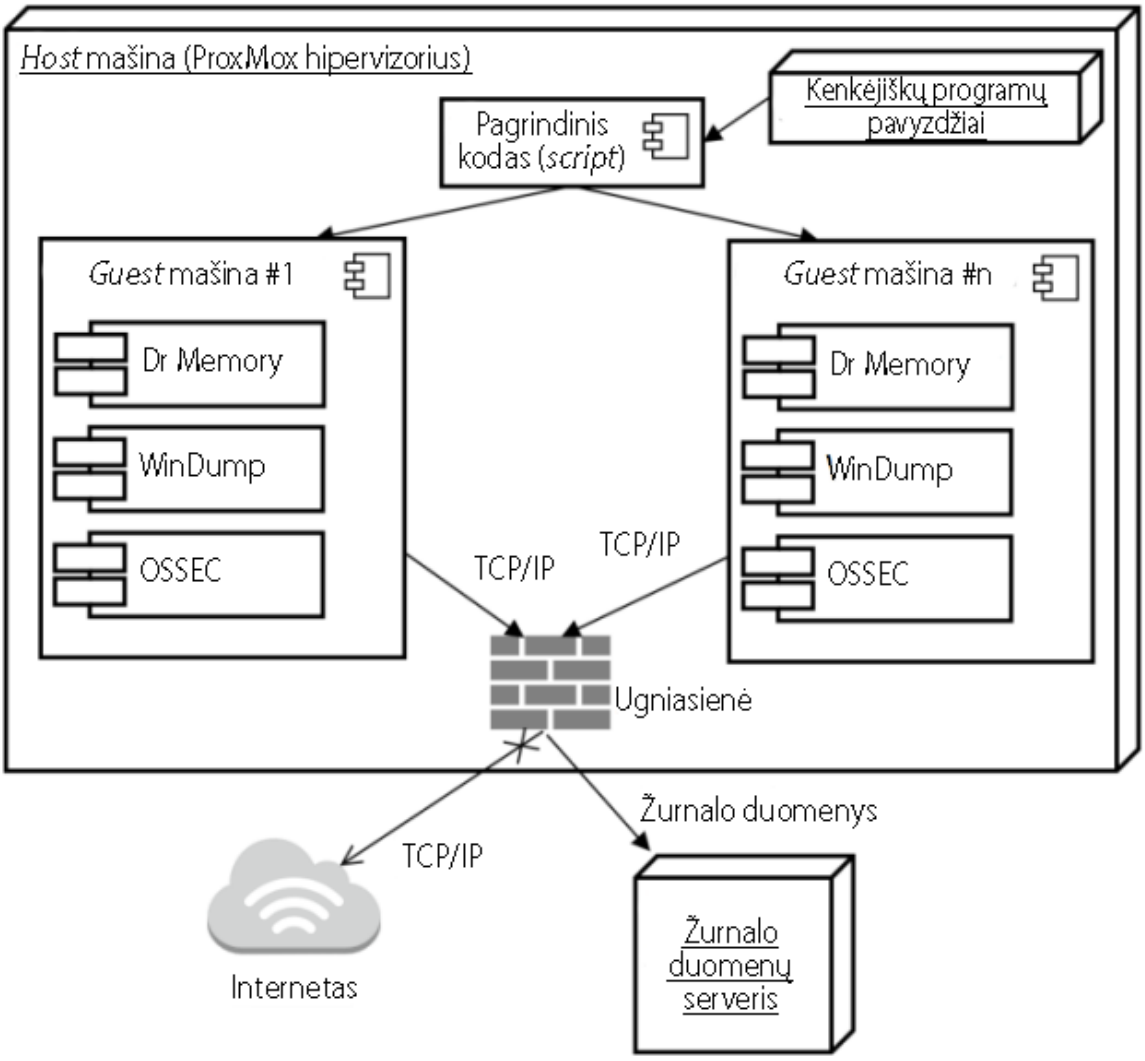
3.1.1. *Metodo detalizavimas*

Čeponio ir Goranin metodo pirminis atraminis taškas yra pagrindinė „bash“ rašmena (angl. *script*), kuri vykdo visas svarbias komandas, reikalingas tyrimo duomenims rinkti. Rašmena paruošia, paleidžia ir sustabdo „Guest“ virtualią mašiną, kurioje įdiegta „Windows 7“ OS. Pagrindinė „bash“ rašmena turi tik vieną, prieš paleidimą, perduodama parametrą – aplankos lokaciją, kurioje yra paruošti KPK pavyzdžiai. „ProxMox“ ugniasienė yra sukonfigūruota ir įjungta „Host“ mašinoje, kad būtų galima valdyti tinklo srautą ir sumažinti KPK plitimo riziką. Buvo leidžiamas tik vienos krypties srautas į nuotolinį HIDS serverį – visi kiti ryšiai buvo užblokuoti. Visi į tą serverį išsiųsti

duomenys buvo saugomi LOG serveryje, kad būtų galima vėliau analizuoti (Čeponis & Goranin, 2018).

Anomalijų duomenys buvo renkami trimis įrankiais: „Dr. Memory“ suteikė sistemos kreipinių sekimą „Windows“ OS, „OSSEC“ – failų vientisumo stebėjimą bei „WinDump“ – su tinklu susijusios informacijos rinkimą. „Dr. Memory“ įrankis pateikia ne tik sistemos kreipinio pavadinimą, bet ir perduodamų parametrų sąrašą bei grąžinimo reikšmę (Čeponis & Goranin, 2018).

Kenkėjiškų programų vykdymo komponentų schema pateikiama sekančioje iliustracijoje 3.1:



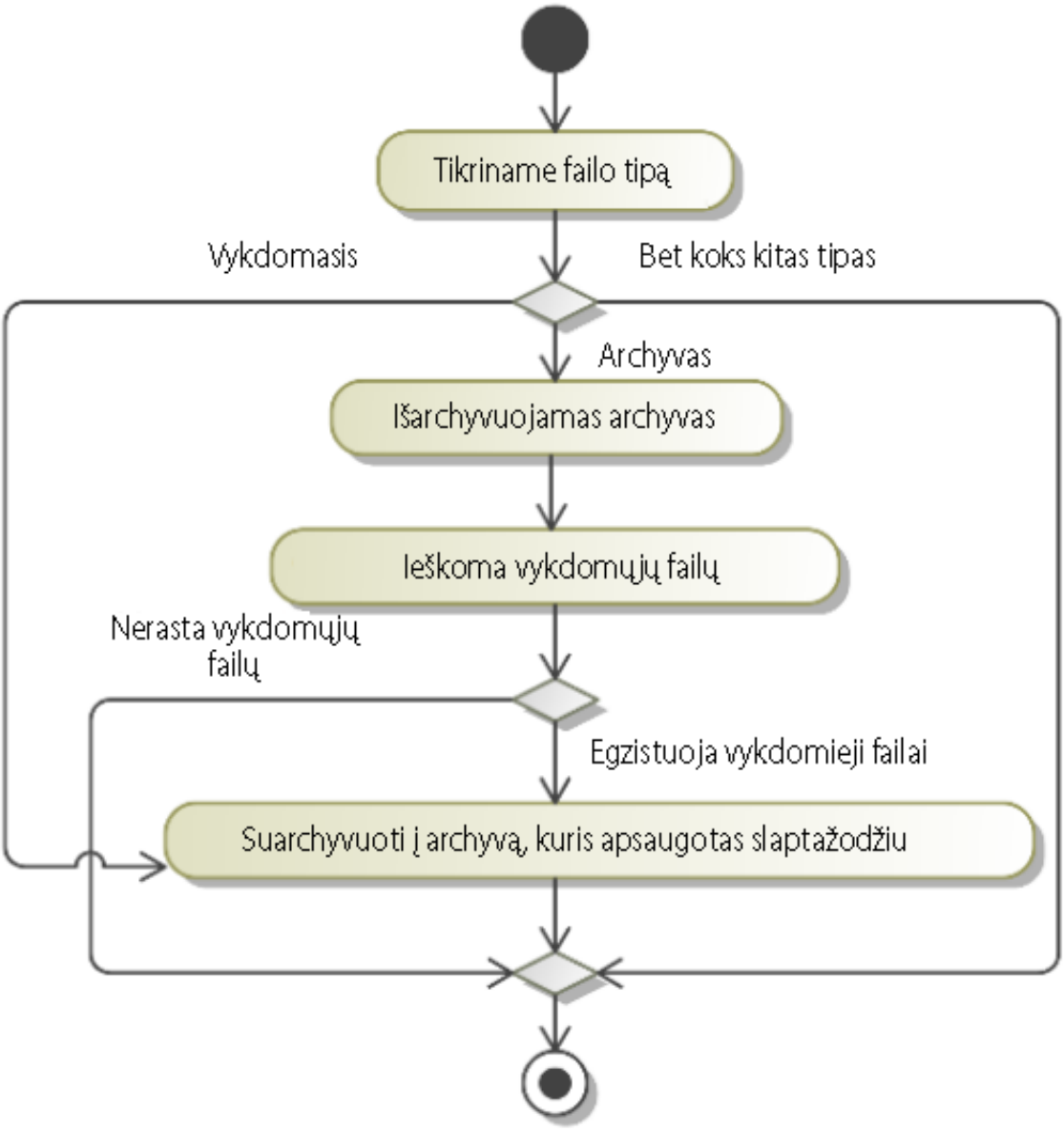
3.1 pav. Kenkėjiškų programų vykdymo komponentų schema (Čeponis & Goranin, 2018)

3.1.2. *Kenkėjiškų programų paruošimas*

Čeponio atliktame tyrime buvo pasirinktas KPK rinkinio „VirusShare_00289.zip“ paketas, kuris yra viešai prieinamas „Virus Share“ duomenų bibliotekoje. Šiame pakete saugomi kenkėjiškų programų archyvai, apsaugoti slaptažodžiu nuo netyčinio panaudojimo. Kiekvienas paketo failo tipas

turėjo būti išanalizuotas, nes failo plėtiniai nebuvo nenurodyti. Kiekviename pakete galėjo būti įvairių tipų kenkėjiškų failų, skirtų paleisti skirtingose OS, pavyzdžiui: „Linux“, „Windows“, „Mac“, „Android“ ir „iOS“. Dėl šios priežasties kiekvienas KPK paketas turėjo būti išanalizuotas ir tyrimui išrinkti tik „Windows“ OS kenkėjiškos programos. „Virus Share“ pavyzdžiai buvo sujungti su „theZoo“ kenkėjiškų programų rinkiniu, kuriame yra populiariausių ir kontraversiškiausių kenkėjiškų programų pavyzdžiai. „theZoo“ duomenų bazėje jau buvo slaptažodžiu apsaugotų archyvų su vykdomaisiais failais. Dėl šios priežasties specialios paruošimo procedūros nereikėjo (Čeponis & Goranin, 2018).

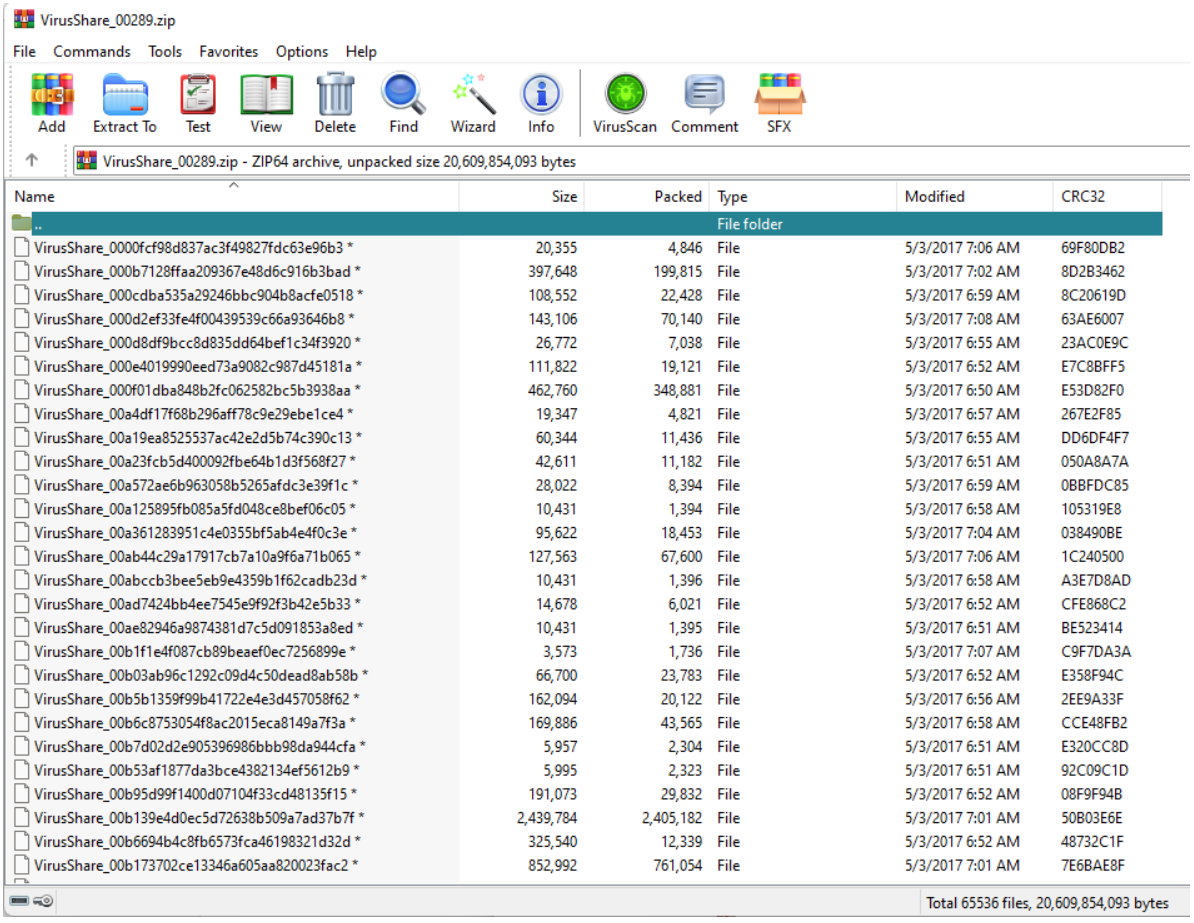
Kenkėjiškų programų pavyzdžių paruošimas parodytas 3.2 pav.:



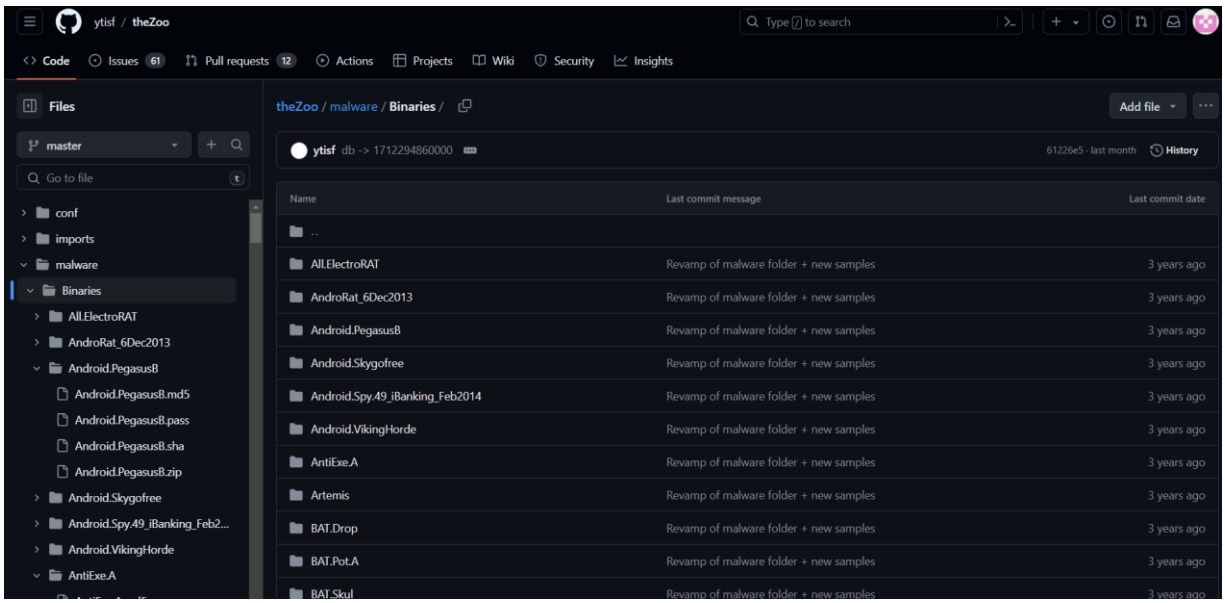
3.2 pav. Kenkėjiškų programų failų paruošimas (Čeponis & Goranin, 2018)

Įprastai pirmame failo baite yra informacija apie failo tipą. Jei tai jau „Windows“ vykdomasis failas – prie jo pridedamas atitinkamas failo plėtinys ir failas supakuojamas į archyvą su slaptažodžiu „infected“. Jei analizuojamas failas yra archyvas – jis išarchyvuojamas tolesnei analizei ir, jei randama vykdomųjų failų, jie pridedami prie slaptažodžiu apsaugoto archyvo. Visi kiti failai praleidžiami (Čeponis & Goranin, 2018).

Kenkėjiškos programinio kodo rinkinys sudaromas sujungiant „VirusShare“ (pav. 3.3) ir „thezoo“ (pav. 3.4) rinkinius bei išfiltruojant tikrąsias kenkėjiškas programas nuo „False-positive“ programų bei failų, kurie vėliau bendruomenių tapo pripažinti kaip nekenkėjiški failai, nekeliantys pavojaus. Atrinkus kenkėjiškas programas, atliekamas papildomas filtravimas, kad išrinkti vykdomąsias programas.



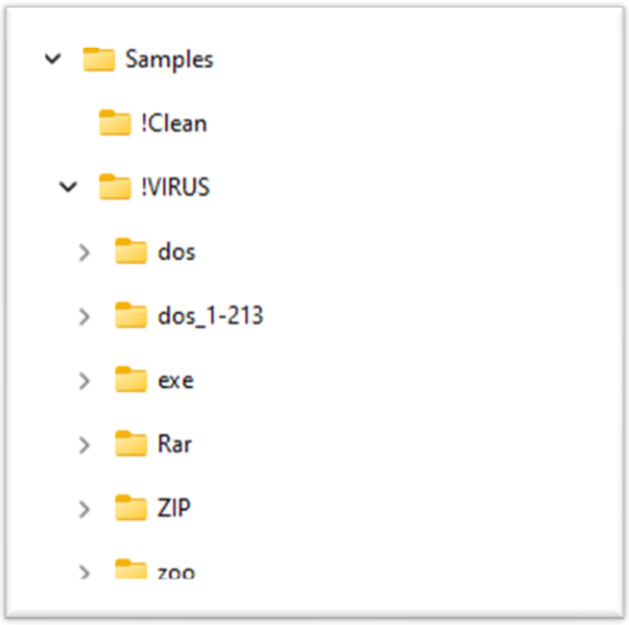
3.3 pav. „VirusShare“ KPK rinkinys



3.4 pav. „theZoo“ KPK rinkinys

Paruoštą rinkinį sudaro (pav. 3.5):

- Nekenksmingi KPK aplankale „!Clean“;
- Atrinkti KPK vykdomųjų failų archyvai, aplankalu „!VIRUS“:
  - 6 aplankalai: „dos“, „dos_1-213“, „exe“, „Rar“, „ZIP“, „zoo“.
  - Vykdomųjų failų archyvai aplankaluose: „dos“, „dos_1-213“, „exe“;
  - Archyviniai (neatlikus analizės – tipas nežinomas) archyvai aplankaluose „Rar“, „ZIP“;
  - Įvairūs KPK iš „thezoo“ rinkinio
  - Įvairūs KPK iš „Virus Share“ duomenų bibliotekos. Naudojamas ankstesniame tyrime taikytas paketas – „VirusShare_00289.zip“



3.5 pav. Paruošto KPK rinkinio struktūra

3.1.3. *Metodo rezultatai*

„Virus Share“ imtį sudaro 12110 suarchyvuoti bei apsaugoti slaptažodžiu vykdomų kenkėjiškų programų pavyzdžiai, kurie buvo panaudoti tyrimo duomenų rinkiniui generuoti. Visi KPK pavyzdžiai buvo testuojami per du mėnesius (2017.07-2017.08) ir 89.34 % (10820) iš jų buvo sėkmingai įvykdyti. Tolesnis patikrinimas atskleidė, kad ne visi gauti kenkėjiškų programų pavyzdžiai gali būti laikomi 100 % kenkėjiškais. Duomenų rinkiniui buvo atrinkti tik pavyzdžiai, turintys 15 ir daugiau teigiamų KPK aptikimų – ši taisyklė leido pasirinkti 10 276 kenkėjiškų programų pavyzdžių iš 10 820 išbandytų (Čeponis & Goranin, 2018).

Sistemos kreipinių pėdsakų rinkimas atliktas naudojant „drStrace“ įrankį. Tai leido surinkti visą reikiamą sistemos kreipinių informaciją: sistemos kreipinio pavadinimą, perduodamų parametrų informaciją (parametrų skaičius ir reikšmes), grąžinimo reikšmes ir vykdymo rezultatą (sėkmingą arba klaidingą) (Čeponis & Goranin, 2018).

Visi sistemos kreipiniai buvo įrašyti į žurnalo failus. Atskiroje C++ programoje buvo įdiegtas specialus žurnalų konvertavimas į geriau apibrėžtus formatus: „JSON“ ir „SQLITE“ duomenų bazės įrašus. Iš viso į duomenų bazę įrašyta 112,56 mln. sistemos kreipinių pėdsakų, sugeneruotų 10276 kenkėjiškų programų pavyzdžių imties. Toks duomenų kiekis turėjo didžiulę įtaką duomenų bazės dydžiui – sugeneruotas „SQLITE“ failas užima 39,1 GB saugyklos vietos (Čeponis & Goranin, 2018).

3.1.4. *Metodo pagerinimo potencialas*

Atskleisti tyrimo rezultatai rodo, kad kone 89.34 % kenkėjiškų programų pavyzdžių buvo sėkmingai ištestuoti. Taikant minėtus įrankius bei analizės duomenų išgavimo būdą, nėra

užtikrinama, kad gaunami rezultatai nebus pažeisti. Duomenų kaupimas ir pirminis išsaugojimas vyksta testuojamoje „Guest“ izoliuotoje aplinkoje, kurioje ir yra paleidžiama kenkėjiška programa. Atsiranda rizika, kad priklausomai nuo kenkėjiškos programos tipo, gali būti pažeisti ne tik testuojamos aplinkos OS kritiniai failai, tačiau ir kaupiamasis failas, kuriame patalpinami analizės duomenys prieš juos perduodant į išorinę saugyklą. Tad grąžinami hipervizoriui rezultatai gali būti sugadinti kenkėjiškos programos. Reikalingas saugesnis būdas, užtikrinantis analizės duomenų saugumą bei integralumą.

Kitas aspektas – tyrimo trukmė. Efektyvumas yra labai svarbus kriterijus tiriant tiek mažos apimties, tiek didelės apimties duomenų masyvus. Tyrimas parodė, kad ištirti 12110 vykdomų kenkėjiškų programų pavyzdžių imtį prireikė dviejų mėnesių periodo. Tad atsiranda reikiamybė šią procedūrą pagreitinti, rasti būdus potencialiai labiau optimizuoti procesus.

Trečiasis aspektas – sukauptų duomenų apimties dydis. Iš ties, išanalizavus tokį didelį kiekį pavyzdžių, nenuostabu, kad išnaudojama net 39,1 GB duomenų bazės saugyklos vietos. Kyla klausimas, ar galima tai optimizuoti, kad būtų išnaudota mažiau vietos. Norint geriau suprasti, duomenų apimties dydį, sekančiame šio darbo skyriuje yra apžvelgiamas būtent šis – AWSCTD duomenų rinkinys.

3.2. „AWSCTD“ duomenų rinkinio analizė

Kenkėjiško programinio kodo atakų sukeltų „Windows“ OS sistemos kreipinių pėdsakų duomenų rinkinys (angl. *Attack-Caused Windows OS System Calls Traces Dataset*) yra sudarytas iš šių sistemos kreipinių, kurie pateikiami tokiu formatu: funkcijos pavadinimas, perduodamų argumentų sąrašas ir jų reikšmės, grąžinama reikšmė bei vykdymo rezultatas. Sistemos kreipinio pavyzdys pateikiamas 3.6 iliustracijoje:

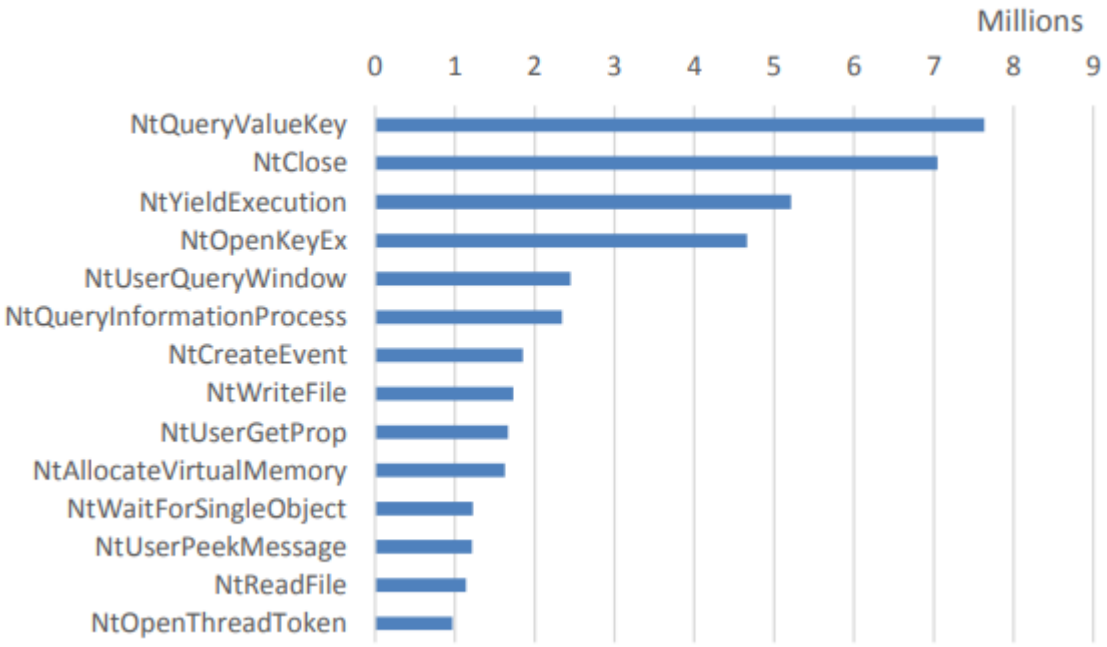
```
NtQueryValueKey
    arg 0: 0x35a (type=HANDLE, size=0x4)
    arg 1: 26/28 "DescriptionID" (type=UNICODE_STRING*, size=0x4)
    arg 2: 0x2 (type=int, size=0x4)
    arg 3: 0x02c5daa0 (type=<struct>*, size=0x4)
    arg 4: 0x90 (type=unsigned int, size=0x4)
    arg 5: 0x02c5da7c (type=unsigned int*, size=0x4)
    failed (error=0xc0000034) =>
    arg 3: <NYI> (type=<struct>*, size=0x4)
    arg 5: 0x02c5da7c => 0x0 (type=unsigned int*, size=0x4)
    retval: 0xc0000034 (type=NTSTATUS, size=0x4)
```

3.6 pav. Sistemos kreipinio pavyzdys (Čeponis & Goranin, 2018)

Tokiame formate sistemos kreipiniai yra žurnalizuojami ir vėliau konvertuojami į „JSON“ bei „SQLITE“ duomenų bazės įrašus. Panaudojus akademines taikomojo programavimo sąsajas, kurias pateikia „Virus Total“, informaciją apie kenkėjiškas programas kas kart sudarė:

- Skanavimo variklis (antivirusinė aplikacija), kuri pateikdavo informaciją apie aptiktą kenkėjišką programą ir jos tipą;
- Teigiama (programa pripažinta kenkėjiška) nuskaitymo rezultatų reikšmė;
- Kenkėjiškos programos aprašymo internetinis puslapis;
- Kenkėjiškos programos elgsenos aprašymas:
    - Failų sistemos veiksmai;
    - Tinklo komunikacija;
    - Įkrauti moduliai („dll“ failai).

„AWSCTD“ duomenų rinkinį sudaro 10 276 ištestuotos kenkėjiškos programos bei 112.56 milijonų surinktų sistemos kreipinių. Dažniausiai pasitaikantys sistemos kreipiniai pavaizduoti 3.7 iliustracijoje:



3.7 pav. Dažniausiai pasitaikantys sisteminiai kreipiniai (Čeponis & Goranin, 2018)

Taigi AWSCTD duomenų rinkinys yra vertingas šaltinis mašininio mokymosi tyrinėtojams ir praktikams, dirbantiems saugos srityje. Duomenų rinkinyje yra sisteminių kreipinių pėdsakai, surinkti iš „Windows“ OS, kurios buvo infekuojamos panaudojus kenkėjiškas programas. Tai yra didelės

apimties duomenų rinkinys, kurį galima naudoti apmokant ir tikrinant mašininio mokymosi modelius, skirtus kenkėjiškų programų aptikimui ir analizei.

Duomenų rinkinys ypač naudingas, nes jame pateikiama daugybė sisteminių kreipinių, iššauktų kenkėjiško programinio kodo. Šiuos duomenys galima naudoti AI modeliams išmokyti atpažinti ir klasifikuoti kenkėjišką veiklą. Be to, duomenų rinkinyje yra įvairių kenkėjiškų programų tipų ir šeimų sistemos kreipinių pėdsakai, todėl galima parengti modelius, galinčius aptikti ir analizuoti daugybę kenkėjiškų programų.

Taip pat verta paminėti, kad duomenų rinkinys yra neapdoroto formato, todėl tyrėjai gali naudoti skirtingus išankstinio apdorojimo metodus, funkcijų išgavimo metodus ir mašininio mokymosi algoritmus duomenims analizuoti.

4. Dinaminės KPK elgsenos rinkimo pagerinimas panaudojant VMI

Siūlomas metodo pagerinimo modelis naudojant VMI

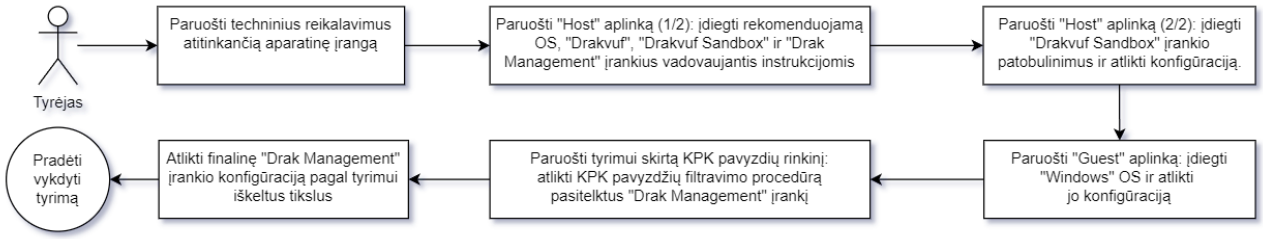
Dinaminės KPK elgsenos rinkimo pagerinimas numatytas pritaikant „Virtual Machine Introspection“ (VMI) technika. Ankstesniuose šio darbo skyreliuose buvo atskleisti VMI pranašumai. Panaudojus „Drakvuf“ bei „Drakvuf Sandbox“ įrankius, kuriuose taikoma VMI technika, šiame darbe buvo sukurtas naujasis dinaminės KPK elgsenos rinkimo metodas. Sukurtas „Drak Management“ įrankis, kuris apjungia „Drakvuf“ bei „Drakvuf Sandbox“ įrankius, dinamiškai automatizuoja bei valdo procesus, apdoroja bei atlieka gautų rezultatų analizes. „Drak Management“ yra plačiai konfigūruojamas įrankis, kuris leidžia tyrėjui keisti daugybę parametrų, norit pasiekti optimaliausią rezultatą.

Atlikus „Drakvuf“ mokslinės literatūros analizę ir D. Čeponio „DrStrace“ su „ProxMox“ įgyvendinto metodo analizę, galima teigti, jog panaudojus VMI technika pasižyminčius įrankius, įžvelgiamas potencialas praplėsti AWSCTD duomenų rinkinį pritaikant kitokią „Windows“ sisteminių kreipinių rinkimo techniką.

Šiame skyriuje apžvelgiamas naujasis dinaminės KPK elgsenos rinkimo pagerinimas:

- Susipažįstama su „Drakvuf Sandbox“ įrankiu, jo veikimo principais.
- Pristatomas sukurtas „Drak Management“ įrankis, kurio pagalba yra apjungiami prieš tai minėti įrankiai.
- Dokumentuojama naujojo įrankio naudojimo instrukcija bei visi pasiruošimo procedūros žingsniai sėkmingo tyrimo analizės rezultatams gauti.

Siekiant geriau suprasti, kaip turėtų būti atliekamas pasiruošimas dinaminės KPK elgsenos rinkimo tyrimui, pateikiama procedūrų seka (pav. 4.1):



4.1 pav. Pasiruošimo planas dinaminės KPK elgsenos rinkimo tyrimui

4.1. „Drakvuf Sandbox“ panaudojimas

„Drakvuf Sandbox“ – tai virtualios aplinkos analizės įrankis, pastatytas ant „Drakvuf“ pamatų, naudojamas KPK analizei. Skirtumas tarp „Cuckoo Sandbox“ ir „Drakvuf Sandbox“ yra tas, kad „Drakvuf Sandbox“ pritaiko VMI techniką, kuri yra ženkliai pažangesnė už tradicinius metodus. Tiek „Drakvuf“, tiek „Drakvuf Sandbox“ yra nuolatos tobulinami, tad svarbu pabrėžti, kad šio darbo rašymo metu, šių įrankių galimybės yra nustatytos ir įvertintos. Todėl kitame skyriuje yra pristatomas naujasis „Drak Management“ įrankis, kuris praplečia minėtų įrankių funkcionalumą.

Prieš pereinant prie „Drak Management“ techninio sprendimo, svarbu suprasti, kaip veikia fundamentiniai dinaminės KPK elgsenos rinkimo komponentai, be kurių „Drak Management“ negalėtų funkcionuoti. Todėl šiame skyrelyje skaitytojas supažindinamas su „Drakvuf Sandbox“ įrankiu bei tam tikrų funkcijų panaudojimu šio rašto darbo pagrindiniam tikslui pasiekti – pagerinti dinaminės KPK elgsenos surinkimą.

Pirmiausia pateikiami aparatinės įrangos bei „Host“ OS reikalavimai atsižvelgiant į tai, kad bus diegiamas „Drak Management“ įrankis.

4.1.1. *Aparatinės įrangos bei „Host“ OS reikalavimai*

Aparatinės įrangos reikalavimai:

- „Intel“ procesorius su virtualizacijos palaikymu (VT-x) ir išplėstinėmis puslapių lentelėmis (EPT). **„Drakvuf“ neveiks jokiuose kituose procesoriuose, pavyzdžiui „AMD“ arba „Intel“ procesoriuose be reikalingų virtualizacijos plėtinių.**
- 1 „Guest“ aplinkai reikalingas mažiausiai 4 branduolių „x64“ architektūros „Intel“ gamintojo procesorius. 2-4 „Guest“ aplinkoms – 6 branduolių procesorius.

- 8 GB operatyvios atminties (RAM) 1 „Guest“ aplinkai. 12-16 GB – 2 „Guest“ aplinkoms.
- 1000 KPK pavyzdžių tyrimui, vykdant 60 sekundžių trukmės analizę ir renkant tik „Windows“ sisteminius kreipinius, rekomenduojama apie 200 GB laisvos kietojo disko atminties.

„Host“ OS reikalavimai:

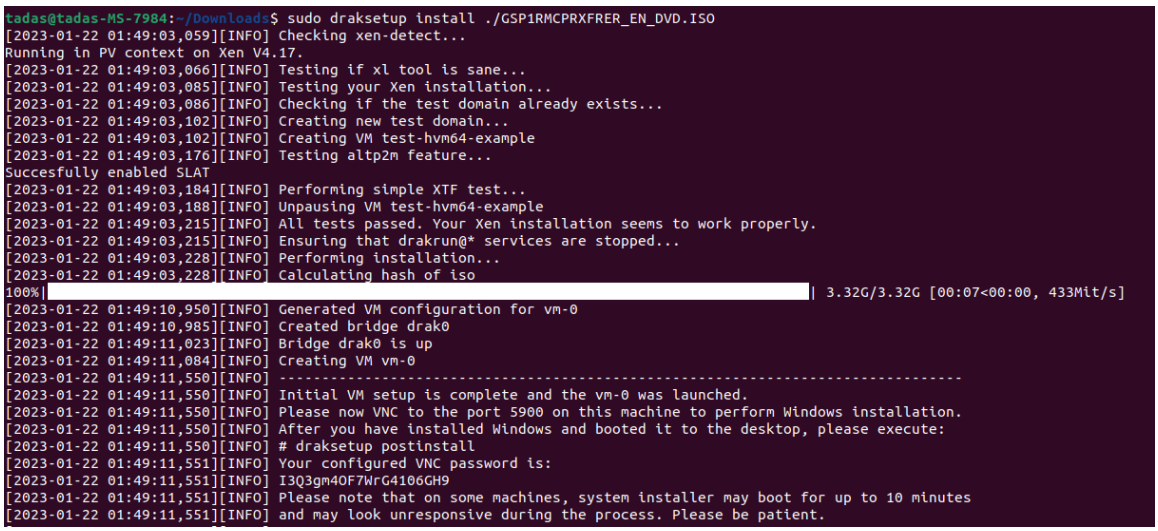
- Ne senesnė, nei „Ubuntu“ 20.04 versija
- Ne senesnis, nei „5.10.0-1019-oem“ branduolys

4.1.2. *„Guest“ VM aplinka*

Atlikus įrankių diegimą, svarbu paruošti „Guest“ VM aplinką. Tad panaudojama atitinkamos „Drakvuf Sandbox“ komandos, kad įdiegti „Windows“ iš „ISO“ disko.

„draksetup [parinktys] [parametrai]“ komanda leidžia automatizuoti OS VM kūrimą. Naudojamos parinktys – 2: „install“ ir „postinstall“. Pirmoji atlieka pirminį „Guest“ aplinkos paruošimą, kuomet vykdomas OS diegimas, kuris gali būti rankinis arba automatizuotas. Rekomenduojama OS diegimą atlikti rankiniu būdų, nes po sėkmingo OS diegimo rekomenduojama atitinkamai sukonfigūruoti OS (apie tai bus aptariama vėliau). Antroji atlieka „Guest“ aplinkos paruošimą tyrimams.

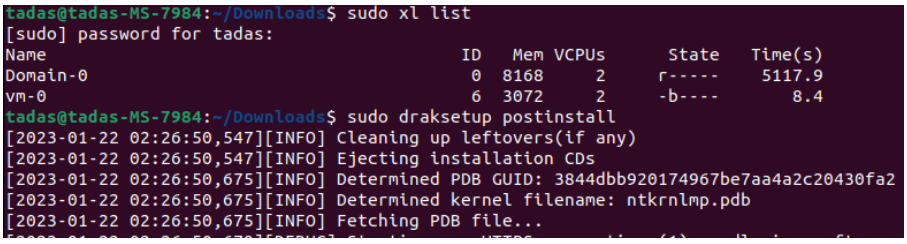
Pirmame pavyzdyje (pav. 4.2) vykdoma „draksetup install windows7diskas.iso“, kuriame atliekamas „Windows 7“ OS diegimas su numatyta konfigūracija. Konfigūracijos pavyzdys pridedamas šio darbo **Prieduose (1)**. Taip pat komandinėje eilutėje galima nurodyti papildomus parametrus, pavyzdžiui „--vcpus 2“ ir „--memory 3072“, tokiu atveju „Guest“ aplinka paleidžiama su 2 virtualiais procesoriais ir 3072 MB virtualiosios atminties.



4.2 pav. „draksetup install“ „Windows 7“ OS diegimas

Prisijungus su „VNCviewer“ ir atliktus „Windows“ diegimo procesą iki galo, galima eiti prie sekančio žingsnio. Paruošus OS, atliekamas esamos (švarios) virtualios mašinos atvaizdo išsaugojimas. Sukuriamas failas su „.sav“ plėtiniu. Tai reikalinga tam, kad kiekvieną kartą po KPK pavyzdžio analizės, virtualioji mašina būtų atstatyta į pradinę – švariąją būseną, taip paruošiant ją sekančiai KPK pavyzdžio analizei.

„xl“ komandos pagalba, matome, kad sukurta virtuali mašina veikia. Jos pavadinimas „vm-0“. „Domain-0“ yra „Host“ sistema, kurios sąsają pateikiama žemiau. Tuomet vykdoma komanda „draksetup postinstall“ (pav. 4.3).



4.3 pav. „draksetup postinstall“ einamasis procesas po OS diegimo

Komanda „draksetup postinstall“ naudojama papildomoms konfigūravimo užduotims atlikti po sėkmingo „Guest“ OS konfigūracijos etapo. Paprastai tai apima šiuos veiksmus:

- „Worker“ servisų įjungimas: ijungiami „drakrun“ bei „drak-postprocess“, atitinkamai VM analizės vykdymo ir duomenų apdorojimo servisai. Šie servisai veikia pastoviuoju rėžimu, tad perkrovus „Host“ mašiną, jie paleidžiami automatiškai.
- Tinklo konfigūravimas: nustatomi VM tinklo nustatymai. Tai apima IP adreso, tinklo kaukės ir numatytojo šliuzo konfigūravimą.

- Užkardos (angl. *firewall*) konfigūravimas: komanda nustato VM užkardą, kad būtų galima prisijungti prie „Host“ sistemos.
- Laiko juostos konfigūravimas: nustatoma „Guest“ VM laiko juosta, kad ji atitiktų „Host“ sistemos laiko juostą. Tai būtina siekiant užtikrinti, kad užfiksuotų įvykių laiko žymos būtų tikslios.
- „VNCviewer“ konfigūravimas: sūkuriams tiesioginio prisijungimo į „Guest“ VM slaptažodis.

Sekantis automatizavimo žingsnis yra „Rekall“ profilių kūrimas „Guest“ virtualios mašinos operacinei sistemai (pav. 4.4).



4.4 pav. „Rekall“ profilių gavimas

„Drakvuf Sandbox“ atlieka užklausas į „Microsoft“ simbolių serverius išgauti „Windows“ „DLL“ bibliotekų „PDB“ failus. „PDB“ („Program DataBase“) tai tokie failai, naudojami siekiant palengvinti programų derinimą (angl. *debugging*). Juose dažniausiai talpinama išsami informacija apie sisteminius kreipinius ir jų ryšius, atminties adresus.

„Drakvuf“ naudoja „Rekall Memory Forensic Framework“ profilius, kad galėtų interpretuoti įvairias „Windows“ OS struktūras ir kitus svarbius duomenis, kurie yra būtini VM atminties analizės metu. „Rekall“ yra atvirojo kodo atminties analizės įrankis, kurį galima naudoti norint iš VM atminties išgauti informaciją, pvz., procesų sąrašus, tinklo ryšius ir kitus duomenis.

4.1.3. *„Drakvuf“ konfigūravimas ruošiant daugiau nei vieną „Guest“ VM*

Atlikus diegimą, pagal numatytuosius nustatymus „Drakvuf Sandbox“ vienu metu galės apdoroti vieną KPK pavyzdį. Servisas, kuris vykdo tiriamą analizę, vadinamas „drakrun“.

Jeigu aparatinės įrangos resursų pakanka, galima konfigūruoti „Drakvuf Sandbox“, kad galima būtų analizuoti kelis KPK pavyzdžius vienu metu, sukuriant kelias „Guest“ VM aplinkas. Tai leidžia lygiagrečiai analizuoti kelis KPK pavyzdžius, o tai gali labai pagerinti bendrą sistemos pralaidumą.

Komandinė eilutė „draksetup scale <skaičius>“ leidžia dinamiškai realiuoju laiku padidinti arba sumažinti lygiagrečių VM kiekį. Pavyzdžiui, turime tokią konfigūraciją, kai vienu metu veikia 5 VM. Panaudojus komandą „draksetup scale 2“ , VM nuo „drakrun@3“ iki „drakrun@5“ bus

išjungtos. Jei šių VM analizė vis dar vykdoma, komanda lauks, kol ji bus baigta. Tokia apsauga leidžia VM iki galo ištirti KPK pavyzdį nenutraukiant VM veikimo jos darbo metu. Toks dinamiškumas gali padėti testavimo metu, kuomet ieškomas optimaliausias vienu metu veikiančios „Guest“ VM kiekis pagal aparatinės įrangos našumo apribojimus.

4.1.4. *„Guest“ VM pradinio atvaizdo modifikavimas*

„Drakvuf Sandbox“ suteikia galimybę modifikuoti „Guest“ VM pradinį atvaizdą panaudojus „drakplayground 0“ komanda. Tuomet leidžiama įkelti įvairius failus, prisijungti prie pradinio „Guest“ VM tiesiogiai per „VNCviewer“, atlikti kažkokius „Guest“ OS pakeitimus. Svarbu pabrėžti, kad baigus šią procedūrą, privaloma visus pakeitimus išsaugoti. Tai galima padaryti su „draksetup postinstall“ komanda. Pavyzdžiai pridedami šio darbo **Prieduose (2)**.

4.1.5. *KPK pavyzdžio analizė*

Kai „Drakvuf Sandbox“ analizuoja KPK pavyzdį, įrankis generuoja įvairių tipų išvesties duomenis, pvz., atminties išklotines, žurnalo failus ir kt. Šis įrankis turi daug pagalbinių įrankių, kurie vadinami „plugins“, todėl priklausomai nuo pasirinktų įrankių kiekio, generuojamų duomenų kiekis gali kisti. Šie išvesties failai yra saugomi „MinIO“, kuris veikia kaip objektų saugojimo serveris. „MinIO“ saugo šiuos failus centralizuotoje ir lengvai pasiekiamoje vietoje, todėl lengva gauti duomenis tolesnei analizei.

„Redis“ yra duomenų struktūros saugykla trumpalaikėje atmintyje, kurią „Drakvuf Sandbox“ naudoja sugeneruotiems duomenims saugoti ir nuskaityti apdorojimui. „Redis“ naudojama duomenims saugoti struktūrizuotu formatu, tai palengvina procesus, kai reikia teikti užklausas į serverį ir analizuoti duomenis. „Redis“ gali saugoti duomenis raktų ir reikšmių porose, maišose, sąrašuose, rinkiniuose bei kituose pavidaluose.

„MinIO“ ir „Redis“ įrankiai dirba kartu, kad pateiktų efektyvų ir patikimą saugojimo bei duomenų apdorojimo sprendimą. „MinIO“ saugo išvesties failus centralizuotoje vietoje, o „Redis“ saugo duomenis struktūrizuotu formatu, kurio pagalba galima lengvai juos analizuoti.

Po sėkmingos ar nesėkmingos KPK pavyzdžio analizės, sugeneruojami rezultatai yra talpinami „Host“ OS specifinėse vietose. Pavyzdys pridedamas šio darbo **Prieduose (3)**. Dažniausiai tai yra „/var/lib/drakcore“ „Host“ OS vieta. Daugiau informacijos apie tai galima rasti „Drakvuf Sandbox“ įrankio dokumentacijoje (CERT Polska et al., 2021). Šiame darbe naudojama tik sisteminių kreipinių analizės dalis. Vieno KPK pavyzdžio analizės sisteminiai kreipiniai yra talpinami „syscall.log“ faile. Pavyzdys pridedamas šio darbo **Prieduose (4)**. Šiuos failus „Drakvuf Sandbox“

nuskaito tuomet, kai „Drak Management“ įrankis per API kreipiasi į „Drakvuf Sandbox“ su užklausa sisteminiams kreipiniams išgauti. Detalesnis procesą aprašomas kitame skyriuje.

Galima atkreipti dėmesį, kaip atrodo vieno surinkto sisteminio kreipinio pavyzdys:


```
{"Plugin": "syscall", "TimeStamp": "1674351501.655886", "PID": 1480, "PPID":
3060, "TID": 2392, "UserName": "SessionID", "UserId": 1, "ProcessName":
"\\Device\\HarddiskVolume2\\Users\\admin\\Desktop\\CrystalDiskInfo8_17_14.exe",
"Method": "NtUserMessageCall", "EventUID": "0x2d058", "Module": "win32k",
"vCPU": 1, "CR3": "0x87556000", "Syscall": 7, "NArgs": 7, "Inlined": "False",
"hWnd": "0x40196", "Msg": "0x81", "wParam": "0x0", "lParam": "0x8c840",
"ResultInfo": "0x0", "dwType": "0x29e", "Ansi": "0x0"}
```


Pateikta išvestis yra „JSON“ objektų, kuriuose yra informacijos apie proceso atliekamus sistemos kreipinius, serija. Kiekvienas „JSON“ objektas reiškia vieną sistemos kreipinį ir jame yra įvairių laukų, kuriuose pateikiama išsami informacija apie kreipinį. „JSON“ objektų paaiškinimai eilės tvarka yra pateikiami 4.1 lentelėje.

4.1 lentelė. „syscall.log“ sisteminio kreipinio įrašo objektų paaiškinimai

Objektas	**Paaiškinimas**
„Plugin“	Šis laukas rodo, kad išvestį generuoja „syscall“ papildinys
„TimeStamp“	Šiame lauke yra laiko žyma, kada buvo atliktas sisteminis kreipinys, sekundėmis nuo epochos
„PID“	Proceso identifikacinis numeris
„PPID“	Pirminio (tėvinio) proceso identifikacinis numeris
„TID“	Gijos (angl. *thread*) ID
„UserName“, „UserId“	Šiuose laukuose yra vartotojo, kuris vykdo procesą, vartotojo vardas ir vartotojo identifikacinis numeris
„ProcessName“	Šiame lauke yra proceso, kuris iškvietė sisteminį kreipinį, pavadinimas bei kelias iki jo
„Method“	Šiame lauke yra sisteminio kreipinio pavadinimas
„EventUID“	Šiame lauke yra identifikatorius, kuris unikaliai identifikuoja sisteminio kreipinio iškvietimą.
„Module“	Šiame lauke yra modulio, su kuriuo susietas sisteminis kreipinys, pavadinimas.
„vCPU“	Šiame lauke yra virtualus procesoriaus numeris virtualioje mašinoje, kurioje iškviečiamas sisteminis kreipinys.
„CR3“ („Control Register 3“)	Šiame lauke yra „CR3“ registro reikšmė (šešioliktainis skaičius), kuri nurodo proceso puslapio katalogo (angl. *process page directory*) fizinį adresą atmintyje.
„Syscall“	Šiame lauke yra sisteminio kreipinio numeris.
„NArgs“	Šiame lauke yra su sisteminiu kreipiniu perduotų argumentų (funkcijos kvietimo parametrų) skaičius.
„Inline“	Šis laukas nurodo, ar sisteminis kreipinys buvo įtrauktas, ar ne.
(Kiti laukai)	Toliau sekančiuose laukuose yra pateikiami sisteminio kreipinio kvietimo parametrai „NArgs“, jeigu tokie buvo. Pavyzdžiui failų „handles“, atminties adresai ir prieigos teisės. Šie laukai skirsis priklausomai nuo konkretaus sistemos kreipinio.

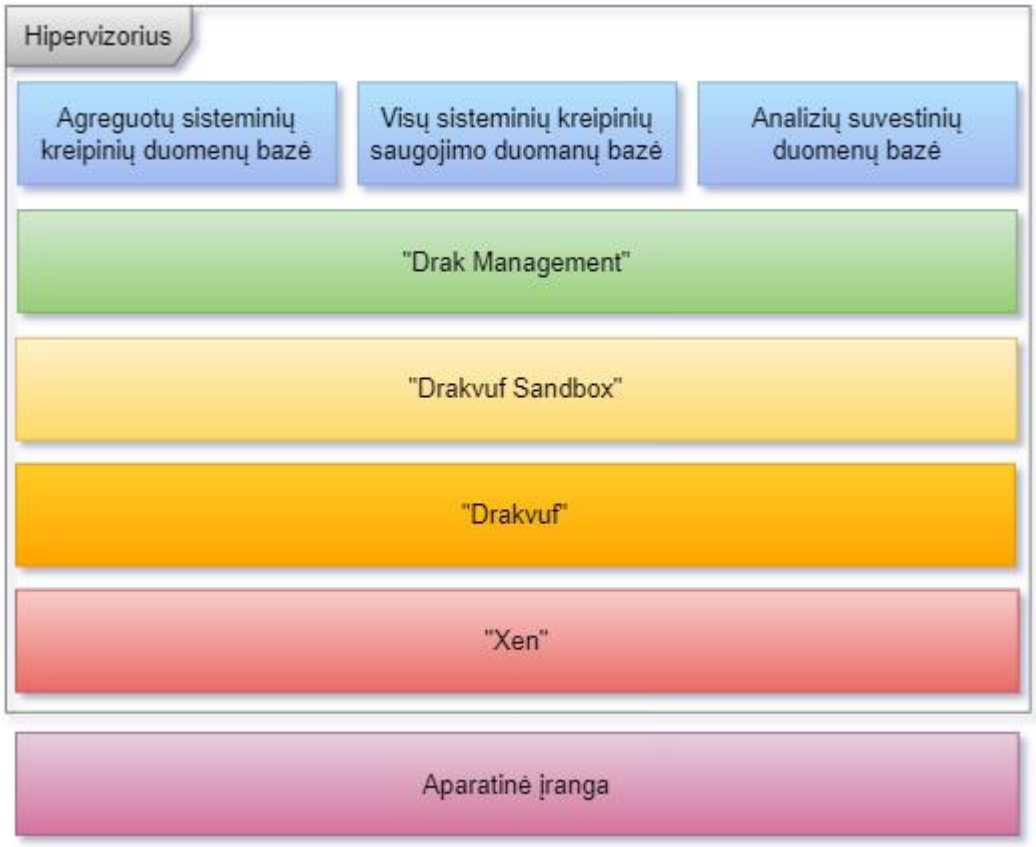
„Drakvuf Sandbox“ gali sugeneruoti įvairių tipų analizės duomenų. Tai priklauso nuo to, kokie pagalbiniai įrankiai buvo pasirinkti prieš pradedant KPK pavyzdžio analizę. Reikia pabrėžti, kad analizė gali drastiškai sulėtėti priklausomai nuo pasirinktų papildomų analizės įrankių kiekio. Pagal nutylėjimą, „Drakvuf Sandbox“ tyrimui pasirenka tam tikrą kiekį analizės įrankių, kurie sugeneruoja tokio tipo analizės failus:

- Įvairių tipų žurnalizuoti failai (įskaitant ir sisteminius kreipinius): šiuose failuose yra išsami informacija apie įvykius, įvykusius analizės metu, pvz., proceso ir gijų kūrimą, prieigą prie failų sistemos ir tinklo veiklą. Ši informacija gali būti naudojama norint suprasti kenkėjiškos programos elgesį ir nustatyti bet kokią kenkėjišką veiklą.
- Atminties išklotinės: šiuose failuose yra VM atminties kopija tam tikru momentu. Šias išklotines galima analizuoti naudojant atminties kriminalistikos įrankius, siekiant gauti papildomos informacijos apie kenkėjišką programą, pvz., jos atminties turinį.
- Ekrano kopijos: „Drakvuf Sandbox“ taip pat gali reguliariais intervalais daryti virtualios mašinos darbalaukio ekrano kopijas, kurios gali būti naudingos norint suprasti kenkėjiškos programos veiksmus analizės metu.
- Ataskaitos: „Drakvuf Sandbox“ gali sugeneruoti ataskaitą, kurioje apibendrinama analizės metu surinkta informacija, pvz., pasiekti failai, modifikuoti „Windows“ registro raktai ir kenkėjiškos programos užmegzti tinklo ryšiai.

Vertėtų paminėti, kad pagrindinis analizės subjektas yra sisteminiai kreipiniai, todėl bus atsižvelgiama tik į tai. Kitame skyriuje apžvelgiamas naujasis „Drak Management“ įrankis, jo sąveikavimas su „Drakvuf Sandbox“ įrankiu ir jo generuojamais analizių duomenimis.

4.2. Naujasis „Drak Management“ įrankis

„Drak Management“ architektūrinis sprendimas įgyvendintas su „Python“ bei „SQLITE 3“ technologijomis. Vieša prieiga prie programinio kodo pateikiama šio darbo literatūros šaltinių sąraše (Švabauskas, 2024) bei kartu su demonstracijos vaizdine sukurtos sistemos medžiaga **Prieduose (6)** bei literatūros šaltinių sąraše (Švabauskas, 2024). Be to, „Drak Management“ negali funkcionuoti be „Drakvuf“ ir „Drakvuf Sandbox“, todėl privaloma užtikrinti, kad minėti įrankiai yra sudiegti bei tinkamai sukonfigūruoti „Host“ sistemoje. Norint geriau suprasti, kaip realizuojamas dinaminės KPK elgsenos pagerinimas, pateikiama architektūrinė sluoksnių diagrama (pav. 4.5).

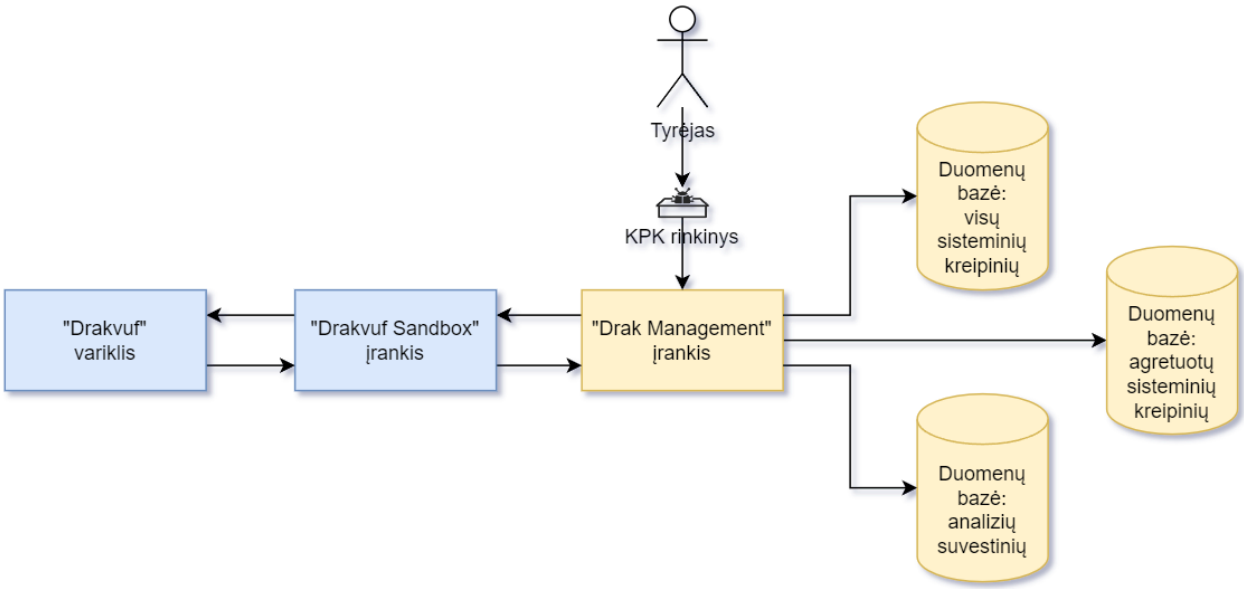


4.5 pav. Architektūrinių sluoksnių diagrama

Šioje iliustracijoje, „Aparatinė įranga“ yra pamatinis sluoksnis, ant kurio konstruojama sistema. „Drakvuf“ analizėje sužinojome, kad hipervizorių tipai yra „hosted“ arba „bare-metal“. Šiuo atveju buvo pasirinkta „bare-metal“ dėl geresnio suderinamumo bei greitaveikos. Aukščiau seka „Xen“ hipervizorius, kuriame veikia „Drakvuf“ variklis. Toliau – „Drakvuf Sandbox“ įrankis, kuris suteikia API, vartotojo sąsają bei vykdo analizę. Toliau – „Drak Management“ įrankis, kuris yra atsakingas už procesų automatizavimą bei gautų analizių duomenų surinkimą bei apdorojimą. Pabaigoje – duomenų bazės, kuriose saugomi išfiltruoti KPK analizių sisteminiai kreipiniai, agreguoti sisteminiai kreipiniai bei analizių suvestinės.

Architektūrinių sluoksnių diagrama įrodo, kad norint užtikrinti „Drak Management“ veikimą, būtina pasirūpinti, kad visi žemiau esantys komponentai yra tinkamai sukonfigūruoti ir paruošti darbui. Atlikti paprastą testavimą su galima ir be „Drak Management“ įrankio – tam užtenka „Drakvuf Sandbox“. Naudojantis fundamentaliomis „Drakvuf Sandbox“ įrankio funkcijomis galima išbandyti sistemą ir jos stabilumą atliekant tyrimą su 1 KPK pavyzdžiu. Gautus testavimo rezultatus galima panaudoti „Drak Management“ sistemos testavimui – tereikia minimaliai modifikuoti „Drak Management“ įrankio programinį kodą. Apie tai daugiau informacijos galima rasti įrankio techninėje dokumentacijoje, kuri pridedama kartu su „Drak Management“ programinės įrangos kodu.

Vartotojas pateikia KPK pavyzdžių rinkinį „Drak Management“ įrankiui, kuris gali atlikti filtravimą ir paruošti tinkamą „Windows“ OS tyrimui rinkinį. „Drak Management“ įrankis glaustai sąveikauja su kitais sisteminiais komponentais, perduoda KPK pavyzdį „Drakvuf Sandbox“ įrankiui ir laukia analizės rezultatų, kuriuos vėliau apdoroja ir talpina atitinkamose duomenų bazėse. Tai vaizduojama komponentų tarpusavio sąveikavimo diagramoje (pav. 4.6).



4.6 pav. Komponentų tarpusavio sąveikavimo diagrama

Šioje diagramoje šviesiai geltona spalva (dešinėje diagramos pusėje) yra atskiriami naujai sukurto „Drak Management“ įrankio komponentai, o šviesiai mėlyna (kairėje diagramos pusėje) – reikalingi įrankiai, kuriuos tyrėjas iš anksto privalo įdiegti bei sukonfigūruoti.

Tyrėjui gali tekti atlikti „Drakvuf Sandbox“ įrankio „sutaisymą“, jei nepavyksta užbaigti „host“ aplinkos OS diegimo procedūros. Reikia pabrėžti, kad „Drakvuf Sandbox“ įrankis yra nuolatos tobulinamas, tad priklausomai nuo ateities tyrimų atlikimo laiko, įrankio taisyti gali ir neprireikti. Sutaisymo instrukcijos ir programinis kodas yra pateikiamas kartu su „Drak Management“ įrankiu.

4.2.2. *Įrankio konfigūracija*

„Drak Management“ įrankio konfigūracija „config.ini“ yra sudaryta iš 7 skilčių: administracinių nustatymų, API, „SQLITE“ duomenų bazės, „Guest“ VM specifikacijos, valdymo, automatizavimo ir failų bei aplankalų lokacijų apibrėžimų. Toliau pateikiamas konkretus pavyzdys ir kintamųjų paaiškinimai. Svarbu atkreipti dėmesį, kad „true“ arba „false“ yra ekvivalentu „taip“ arba „ne“.

Administraciniai nustatymai:

```
[admin]
save_logs = true
bypass_xl_check = false
bypass_drak_healthcheck = false
check_internet_connection_is_disabled = true
sudo_password = tadassv
```

- „save_logs“ – ar norime išsaugoti komandinės eilutės žurnalo istoriją. Pagal nutylėjimą – taip.
- „bypass_xl_check“ – ar norime praleisti „Xen“ hipervizoriaus veikimo patikrą. Pagal nutylėjimą – ne.
- „bypass_drak_healthcheck“ – ar norime praleisti „Drakvuf Sandbox“ servisų gyvybingumo patikrą. Pagal nutylėjimą – ne.
- „check_internet_connection_is_disabled“ – ar norime užtikrinti, kad tyrimas leidžiamas tik tuomet, kai internetas yra išjungtas. Pagal nutylėjimą – taip.
- „sudo_password“ – čia tyrėjas įveda „sudo“ slaptažodį, kad „Drak Management“ galėtų sėkmingai: tikrinti „Xen“ hipervizoriaus veikimą, nesant sukurtoms duomenų bazėms galėtų jas sukurti.

API nustatymai:

```
[api]
api_host = http://localhost:6300
```

- „api_host“ – pateikiama „Drakvuf Sandbox“ API nuoroda. Pagal nutylėjimą – „http://localhost:6300“.

„SQLITE“ nustatymai:

```
[sqlite_database]
hostname = localhost
port = 3306
number_of_attempts_if_failed = 5
initial_delay_between_attempts_if_failed = 3
```

- „hostname“ – pateikiamas „SQLITE“ duomenų bazės adresas, kadangi duomenų bazė veikia vietinėje aplinkoje, rekomenduojama reiškmės nekeisti. Pagal nutylėjimą – „localhost“.
- „port“ – pateikiamas „SQLITE“ duomenų bazės prievadas (angl. *port*). Rekomenduojama jo nekeisti. Pagal nutylėjimą – „3306“.

- „number_of_attempts_if_failed“ – nurodomas limitas, kiek kartų „Drak Management“ gali kreiptis į „Drakvuf Sandbox“ API jeigu iš API gaunamas klaidingas atsakymas, pavyzdžiui, jeigu analizė atlikta, tačiau „Drakvuf Sanbox“ dar nespėjo apdoroti duomenų. Rekomenduojama – mažiausiai 5. Pagal nutylėjimą – 5.
- „initial_delay_between_attempts_if_failed“ – pradinis uždelsimo laikas (sekundės), kuris po kiekvieno bandymo (gavus kladingą atsakyma iš API) yra dvigubinamas. Tad po kiekvieno pakartojimo laukimo laikas ilgėja. Rekomenduojama – mažiausiai 3. Pagal nutylėjimą – 3.

„Guest“ VM nustatymai:

```
[guest_vm_config]
memory_mb = 3076
```

- „memory_mb“ – nurodoma, kiek MB operatyvios atminties skiriama „Guest“ virutaliai mašinai. Pagal nutylėjimą – 3076.

Valdymo nustatymai:

```
[management]
verify_windows_executables = false
number_of_concurrent_guests = 2
delay_between_concurrent_threads_seconds = 3
analysis_time_seconds = 60
# default is "syscalls", however, there are more options:
# [ apimon,bsodmon,clipboardmon,cpuidmon,crashmon,debugmon,delaymon,
# exmon,filedelete,filetracer,librarymon,memdump,procdump,procmon,
# regmon,rpcmon,ssdtmon,syscalls,tlsmon,windowmon,wmimon ]
# example to add more -> analysis_tools = syscalls, memdump, regmon
analysis_tools = syscalls
retry_grace_time_seconds = 1
number_of_retries_per_sample = 3
aggregate_syscalls_after_batch_analysis = true
```

- „verify_windows_executables“ – ar norime prieš pradedant tyrimą patikrinti, ar visi pateikti tyrimui pavyzdžiai yra „Windows“ vykdomieji failai ir yra tinkami analizei. Jei bent vienas yra netinkamas – rodomas klaidos pranešimas, programa stabdoma. Pagal nutylėjimą – ne.
- „number_of_concurrent_guests“ – nurodomas kiekis vienu metu planuojamų paleisti „Guest“ VM. Svabu užtikrinti, kad prieš tai buvo panaudota komanda „draksetup scale N“. N – yra tas skaičius, kuris turi būti įvedamas čia. Pagal nutylėjimą – 2.

- „delay_between_concurrent_threads_seconds“ – pirminis „Guest“ VM uždelsimas tarp paleidimų. Tai reiškia, kad jeigu nurodomos, pavyzdžiui 3 sekundės, o „Guest“ VM kiekis – 2, tuomet pirmoji „Guest“ VM bus paleista iš karto, o antroji „Guest“ VM bus paleista po 3 sekundžių. Pagal nutylėjimą – 3.
- „analysis_time_seconds“ – nurodoma, kiek sekundžių turi trukti vieno KPK pavyzdžio tyrimas. Pagal nutylėjimą – 60.
- „analysis_tools“ – nurodoma, kokius „Drakvuf Sanbox“ pagalbinius įrankius naudoti. Sisteminiams kreipiniams rinkti užtenka „syscalls“. Jei norima pateikti daugiau, rašomas įrankio pavadinimas, tuomet dedamas kabelis, tarpas, ir tuomet rašomas kito įrankio pavadinimas. Pagal nutylėjimą – „syscalls“.
- „retry_grace_time_seconds“ – uždelsimo trukmė (sekundės) tarp nepavykusių bandymų stabilumui užtikrinti. Rekomenduojama – mažiausiai 1. Pagal nutylėjimą – 1.
- „number_of_retries_per_sample“ – nurodoma, kiek kartų leidžiama pakartoti KPK pavyzdžio tyrimą, jeigu tyrimas nepavyko. Jei KPK nepavyko paleisti dėl „Guest“ OS nesuderinamumo su KPK, tuomet analizė nera kartojama. Pagal nutylėjimą – 3.
- „aggregate_syscalls_after_batch_analysis“ – ar norime agreguoti visus surinktus sisteminius kreipinius pasiekus tyrimo pabaigą. Pagal nutylėjimą – taip.

Automatizavimo nustatymai:

```
[automation]
number_of_samples_to_scan = 100
number_of_samples_to_scan_successfully = 100
```

- „number_of_samples_to_scan“ – nurodoma, kiek KPK pavyzdžių išanaliztuoti. Jeigu pavyzdžių mažiau, nei nurodytas skaičius, išanalizuojamas tik tas kiekis, kiek pavyko rasti. Pagal nutylėjimą – 100.
- „number_of_samples_to_scan_successfully“ – nurodoma, kiek KPK pavyzdžių išanalizuoti sėkmingai. Sėkmingai išanalizavus nurodytą kiekį KPK pavyzdžių, programa stabdoma, o likę KPK pavyzdžiai nėra analizuojami. Pagal nutylėjimą – 100.

Failų bei aplankalų lokacijų nustatymai:

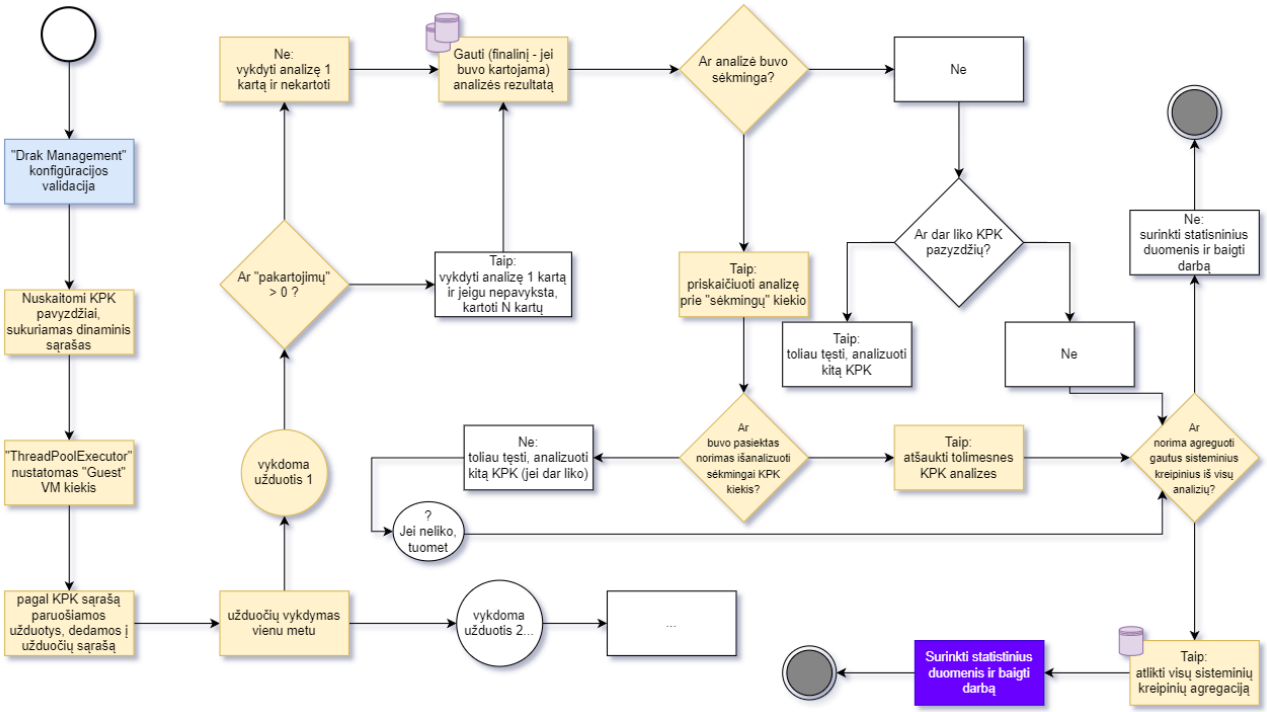
```
[directories]
output_logs_location_file = /home/tadas/Documents/console_output.log
minio_keys_location_file = /etc/drakcore/minio.env
```

```
sqlite_output_db_location_file = /home/tadas/Documents/output/database.db
sqlite_session_output_db_location_file = /home/tadas/Documents/output/session.db
sqlite_aggregated_syscall_db_location_file =
/home/tadas/Documents/output/aggregated.db
malware_samples_location = /home/tadas/Documents/analysis_samples/sorted1stB
```

- „output_logs_location_file“ – nurodoma vieta sistemoje, kur išsaugoti komandinės eilutės žurnalo istoriją. Pagal nutylėjimą – „/home/OSvartotojas/Documents/console_output.log“.
- „minio_keys_location_file“ – nurodoma failo vieta sistemoje, kur gali būti patalpitni „MiniO“ serverio saugos konfigūracija. Rekomenduojama nekeisti. Pagal nutylėjimą – „/etc/drakcore/minio.env“.
- „sqlite_output_db_location_file“ – nurodoma failo vieta sistemoje, kur gali būti sukurtas „SQLITE“ duomenų bazės failas, visiems sisteminiams kreipiniams kaupti. Pagal nutylėjimą – „/home/OSvartotojas/Documents/database.db“.
- „sqlite_session_output_db_location_file“ – nurodoma failo vieta sistemoje, kur gali būti sukurtas „SQLITE“ duomenų bazės failas, visų analizių suvestinėms kaupti. Pagal nutylėjimą – „/home/OSvartotojas/Documents/session.db“.
- „sqlite_aggregated_syscall_db_location_file“ – nurodoma failo vieta sistemoje, kur gali būti sukurtas „SQLITE“ duomenų bazės failas, visiems agreguotiems sisteminiams kreipiniams kaupti. Pagal nutylėjimą – „/home/OSvartotojas/Documents/aggregated.db“.
- „malware_samples_location“ – nurodoma vieta sistemoje, iš kurios imti KPK pavyzdžius tyrimui. Pagal nutylėjimą – „/home/OSvartotojas/Documents/analysis_samples“.

4.2.3. *Įrankio architektūra*

Po „Drak Management“ įrankio konfigūracijos užtikrinus, kad KPK pavyzdžiai yra paruošti, galima pradėti tyrimą paleidus „app.py“ „Python“ rašmeną. Paleista programa pirmiausia atlieka įvairius patikrinimus, pavyzdžiui: tikrinama, ar konfigūracijoje nėra klaidų, ar įrankiui pavyksta komunikuoti su kitais svarbiais komponentais. Jeigu patikrinimai sėkmingi, pradedamas pasiruošimas – imami KPK pavyzdžiai, formuojamas užduočių sąrašas, konfigūruojamas vienu metu veikiančių „Guest“ VM kiekis. Galiausiai pradedamas užduočių vykdymas. „Drak Management“ veiklos diagrama pateikiama 4.7 iliustracijoje.

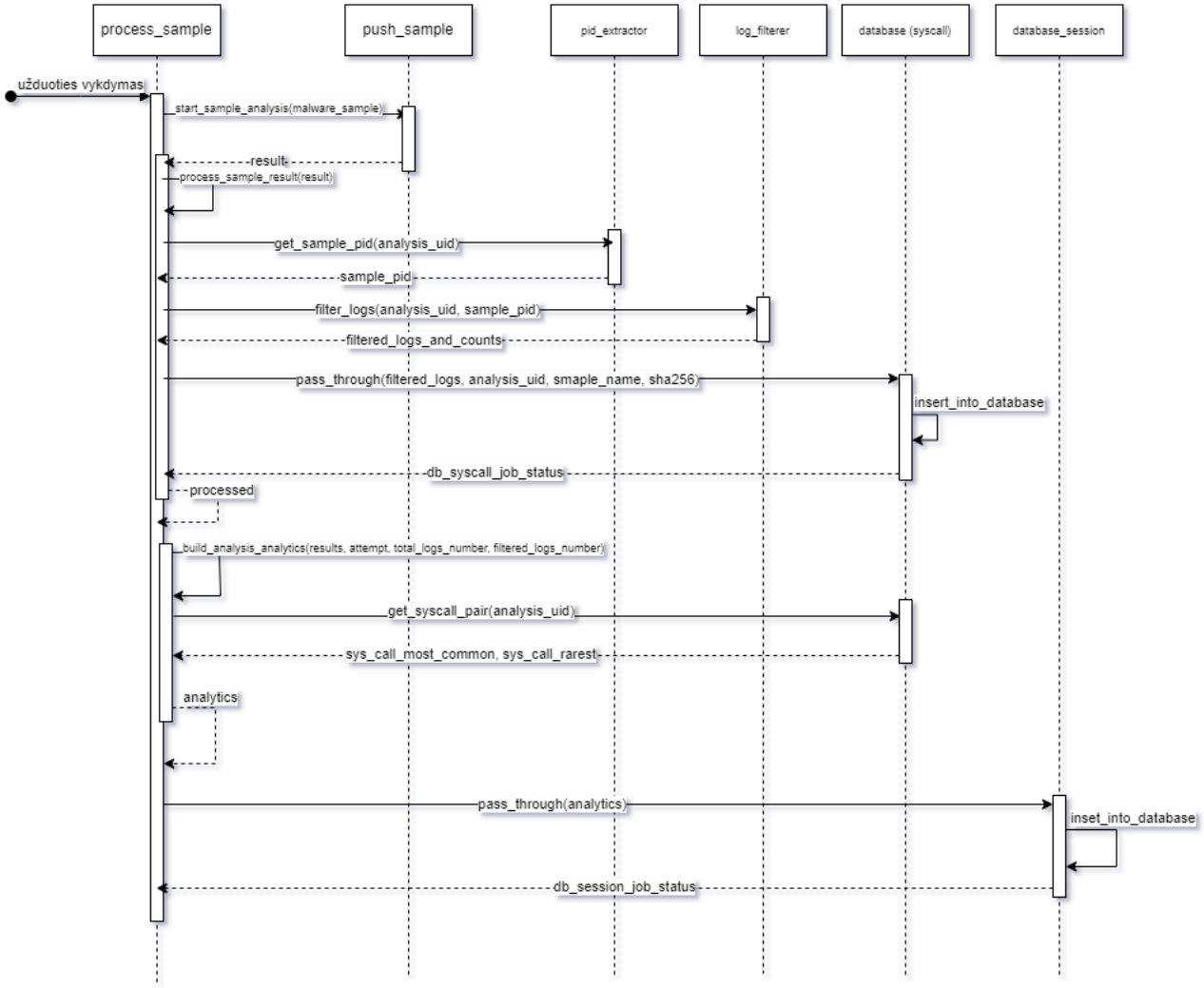


4.7 pav. „Drak Management“ veiklos diagrama (supaprastinta)

Kadangi procesai yra sudėtingi bei priklausantys nuo daugybės kintamųjų, pateiktoje schemoje matoma supaprastinta vykdymo procedūra nuo pat „Drak Management“ įrankio paleidimo iki pat tyrimo pabaigos. Sekama viena iš daugybės užduočių pavyzdžių, kuriame matyti, kad kai pradedamas užduoties vykdymas, atliekamas patikrinimas analizės pakartojimų kiekio nustatymui. Tuomet vykdomas užklausos į „Drakvuf Sandbox“ perdavimas per API. Po „Drakvuf Sandbox“ analizės gaunamas rezultatas ir vyksta patikrinimas, ar analizė buvo sėkminga bei surankami analizės duomenys ir talpinami atitinkamoje duomenų bazėje. Darome prielaidą, kad analizė buvo sėkminga, tuomet vykdomas kitas patikrinimas, ar buvo pasiektas norimas išanalizuoti sėkmingai KPK kiekis. Daroma prielaidą, kad pasiektas. Tuomet atšaukiamos tolimesnės KPK analizės (jeigu dar buvo analizei paruošų uždavinių) ir vykdomas kitas patikrinimas. Patikrinimo metu norima atsakyti į klausimą, ar norima agreguoti visus gautus sisteminius kreipinius. Darome prielaidą, kad atsakymas – taip. Tuomet atliekama visų sisteminių kreipinių agregavimo procedūra. Procedūrai pabaigus, surenkami duomenys ir talpinami duomenų bazėje.

Tuo atveju, jei analizė buvo nesėkminga, tyrėjas yra informuojamas apie tai ir programa tęsia darbą tikrindama, ar dar liko užduočių. Jei liko – toliau tęsiamas darbas – pradedama vykdyti kita užduotis ir ciklas kartojamas. Jei neliko – tuomet pereinama prie finalinio patikrinimo, ar norima agreguoti visus gautus sisteminius kreipinius.

Toliau pateikiama sekų diagrama, kurioje vaizduojama vidinė procesų seka, kai yra analizuojamas specifinis KPK pavyzdys (pav. 4.8).



4.8 pav. „Drak Management“ vieno KPK pavyzdžio analizės sąveikavimo su duomenų baze sekų diagrama

Po to, kai nustatytas analizės pakartojimo kiekis, KPK pavyzdys yra perduodamas apdorojimui. Ši sekų diagrama demonstruoja vidinius analizės procesus.

Taigi, KPK pavyzdys yra perduodamas „Drakvuf Sandbox“ įrankiui per API, kaip ir buvo minėta. Po perdavimo, kas 10 sekundžių vykdoma užklausa į „Drakvuf Sandbox“ rezultatams gauti. Kai analizė baigiama, „Drakvuf Sandbox“ grąžina žurnalizuotus analizės duomenis. „Drak Management“ surenka ir transformuoja juos į vientisą rezultato masyvą. Toliau „Drak Management“ atlieka užklausas įvairiems duomenims gauti, pavyzdžiui, proceso identifikaciniam numeriui, sisteminiams kreipiniams. Gavus duomenis, atliekamas sisteminių kreipinių filtravimas, jų apdorojimas ir talpinimas į duomenų bazę. Tuomet konstruojama analizės rezultato suvestinė – visa analizės informacija apjungiama į vieną įrašą ir išsaugoma domenų bazėje.

Analizės suvestinės konstravimas yra vykdomas gavus analizės rezultatus (pav. 4.9).



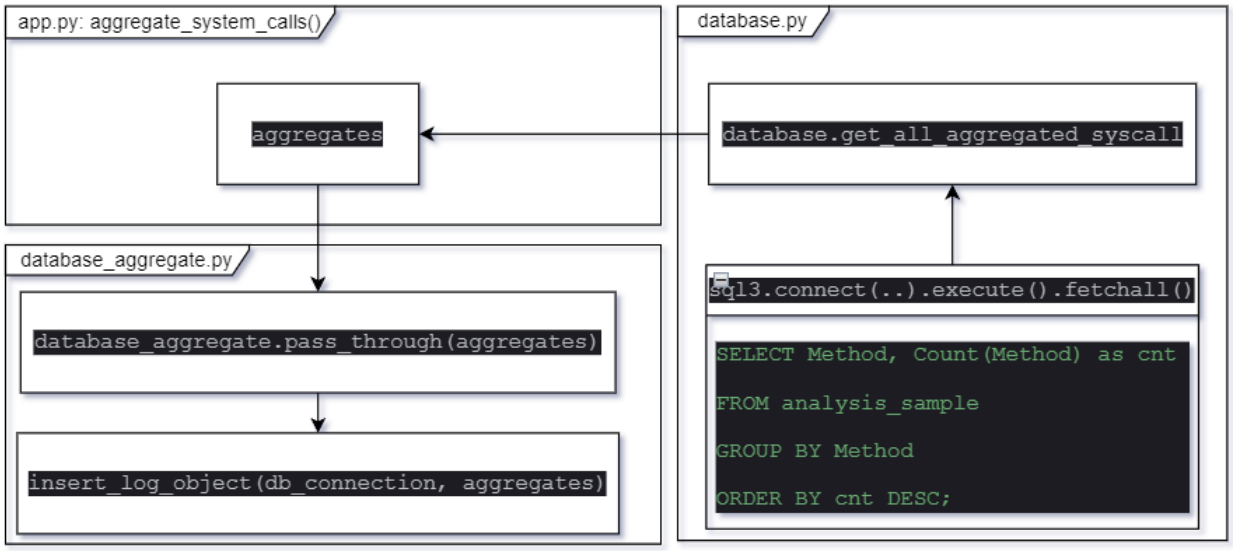
4.9 pav. „Drak Management“ KPK pavyzdžio analizės suvestinės rašymas į duomenų bazę

Čia gauti analizės rezultatai yra transformuojami į naują suvestinės formatą, kuris savyje talpina informaciją apie einamąjį bandymą. Bandymas gali turėti 3 baigtis – „Pass“, „Fail“ ir „Conflict“. Pastaroji reiškia, kad analizė galimai buvo arba sėkminga, arba nesėkminga. Sėkmės atžvilgiu, įvyko komponentų konfliktas, kurio pasėkoje analizės metu išgauti sisteminiai kreipiniai nebuvo apdoroti arba išfiltruoti arba įrašyti į duomenų bazę. Nesėkmės atžvilgiu, analizė neįvyko, nes KPK pavyzdys nebuvo sėkmingai paleistas, o „Drakvuf“ negalėjo klasifikuoti nesėkmingo KPK paleidimo priežasties.

Analizės suvestinė savyje turi ir dažniausiai, ir rečiausiai pasitaikančių sisteminių kreipinių pavadinimus. Šiai informacijai išgauti, vykdomos atitinkamos užklausos į išfiltruotų sisteminių

kreipinių duomenų bazę. Sėkmingai sukonstruota KPK pavyzdžio analizės suvestinė perduodama „session“ duomenų bazei. „Database lock“ problemai išspręsti buvo realizuotas pakartojimo funkcionalumas. Progresyviai didinant uždelsimą tarp bandymų, užtikrinama, kad po tam tikro laukimo, duomenų bazė taps prieinama ir kitas procesas galės atlikti duomenų įrašymą.

„Drak Management“ baigus vykdytas visas užduotis arba kai pasiektas norimas rezultatas, vykdomas patikrinimas, ar norima agreguoti visus surinktus filtruotus sisteminius kreipinius. Jei taip, tuomet atliekamas agregavimo procesas (pav. 4.10).

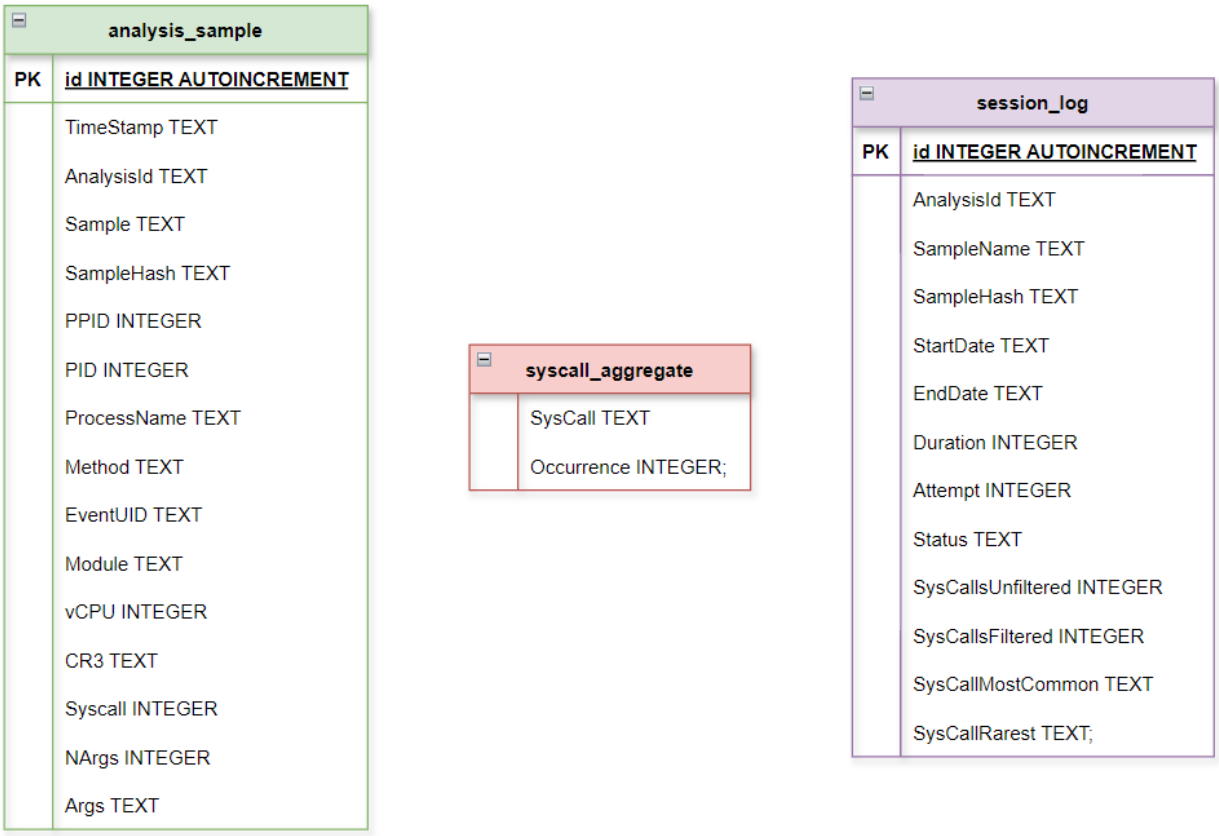


4.10 pav. „Drak Management“ sisteminių kreipinių agregavimo schema (supaprastinta)

Agregavimo metu „Drak Management“ kreipiasi į duomenų bazę, kurioje yra visi filtruoti sisteminiai kreipiniai. Specifinės užklausos dėka, sisteminiai kreipiniai yra agreguojami – grupuojami pagal sisteminio kreipinio pavadinimą ir skaičiuojamas jų dažnis. Rezultatas yra perduodamas kitam komponentui, kuris atlieka gautų duomenų apdorojimą bei išsaugojimą duomenų bazėje. Agregavimo procedūros trukmė priklauso nuo surinktų sisteminių kreipinių kiekio ir aparatinės įrangos galingumo.

4.2.1. *Duomenų bazių realizacija*

Sisteminių kreipinių talpinimui, sisteminių kreipinių agregavimui bei kiekvieno KPK tyrimo rezultatams išsaugoti realizuotos 3 specifinės duomenų bazės. Naudojama technologija – SQLITE. Duomenų bazės struktūra pateikiama iliustracijoje (pav. 4.11).



4.11 pav. Duomenų bazės struktūra

Tai leidžia izoliuoti duomenų bazes efektyvesniam duomenų rašymui, kuomet tyrimai yra vykdomi su didesniu kiekiu vienu metu veikiančių „Guest“ VM.

„analysis_sample“ duomenų bazėje yra talpinami visi išfiltruoti pagal KPK tėvinį bei dukterinį procesus sisteminiai kreipiniai. Struktūra pateikiama 4.2 lentelėje.

4.2 lentelė. „analysis_sample“ duomenų bazės struktūros lentelė

Objektas	**Paaiškinimas**
„id“	Eilutės identifikacinis numeris bei lentelės įrašo pirminis raktas
„TimeStamp“	Laiko žyma, kada buvo atliktas sisteminis kreipinys, sekundėmis nuo epochos
„AnalysisId“	Unikalus analizės identifikacijos numeris
„Sample“	KPK pavyzdžio failo pavadinimas
„SampleHash“	KPK pavyzdžio failo „SHA256“ maiša
„PPID“	Pirminio (tėvinio) proceso identifikacijos numeris
„PID“	Proceso identifikacijos numeris
„ProcessName“	Proceso, kuris iškvietė sisteminį kreipinį, pavadinimas bei kelias iki jo
„Method“	Sisteminio kreipinio, kuris buvo atliktas, pavadinimas
„EventUID“	Identifikatorius, kuris unikaliai identifikuoja sisteminį kreipinį.
„Module“	Modulio, su kuriuo susietas sisteminis kreipinys, pavadinimas.

„vCPU“	Virtualus procesoriaus numeris virtualioje mašinoje, kurioje iškviečiamas sisteminis kreipinys.
„CR3“ („Control Register 3“)	„CR3“ registro reikšmė (šešioliktainis skaičius), kuri nurodo proceso puslapio katalogo (angl. *process page directory*) fizinį adresą atmintyje.
„Syscall“	Šiame lauke yra sisteminio kreipinio numeris.
„NArgs“	Šiame lauke yra sistemos iškvietimui perduotų argumentų (funkcijos kvietimo parametrų) skaičius.
„Args“	Sisteminio kreipinio parametrai, jeigu tokie buvo. Pavyzdžiui failų „handles“, atminties adresai ir prieigos teisės. Šie laukai skirsis priklausomai nuo konkretaus sisteminio kreipinio.

„syscall_aggregate“ duomenų bazėje yra talpinami visi išfiltruoti ir agreguoti sisteminiai kreipiniai, ir galutinis jų aptikimo dažnis. Struktūra pateikiama 4.3 lentelėje.

4.3 lentelė. „syscall_aggregate“ duomenų bazės struktūros lentelė

Objektas	**Paaiškinimas**
„SysCall“	Sisteminio kreipinio, kuris buvo atliktas, pavadinimas
„Occurrence“	Sisteminio kreipinio dažnis (kiek kartų aptiktas)

„analysis_sample“ duomenų bazėje yra talpinami kiekvienos analizės statistiniai duomenys su analizės rezultatu. Struktūra pateikiama 4.4 lentelėje.

4.4 lentelė. „session_log“ duomenų bazės struktūros lentelė

Objektas	**Paaiškinimas**
„id“	Eilutės identifikacinis numeris bei lentelės įrašo pirminis raktas
„AnalysisId“	Unikalus analizės identifikacijos numeris
„SampleName“	KPK pavyzdžio failo pavadinimas
„SampleHash“	KPK pavyzdžio failo „SHA256“ maiša
„StartDate“	Analizės pradžios data
„EndDate“	Analizės pabaigos data
„Duration“	Analizės trukmė (sekundėmis)
„Attempt“	Bandymo numeris
„Status“	Analizės rezultatas („Pass“ / „Fail“ / „Conflict“)
„SysCallsUnfiltered“	Analizės nefiltruotų sisteminių kreipinių kiekis
„SysCallsFiltered“	Analizės filtruotų sisteminių kreipinių kiekis
„SysCallMostCommon“	Dažniausiai aptinkamas sisteminis kreipinys
„SysCallRarest“	Rečiausiai aptinkamas sisteminis kreipinys

Analizių rezultatai gali būti:

- „Pass“ – analizė sėkminga.
- „Fail“ – analizė nesėkminga. Jei „attempt“ lygi 1, tai reiškia, kad nepavyko paleisti KPK pavyzdžio. Jei „attempt“ >1, tai reiškia, kad „Drakvuf“ nepavyko įkelti KPK pavyzdžio į „Guest“ OS dėl kažkokių priežasčių.

- „Conflict“ – sutrikimas, analizė galimai nesėkminga. Tai gali įvykti keliais scenarijais:
    - KPK pavyzdys buvo įkeltas, bet nepavyko jo paleisti ir „Drakvuf“ negalėjo klasifikuoti nesėkmingo KPK paleidimo priežasties.
    - KPK pavyzdys buvo įkeltas ir paleistas – analizė įvyko, tačiau „Drak Management“ įrankiui nepavyko surinkti sisteminių kreipinių.

4.3. Pilnas „Host“ aplinkos paruošimas dinaminės KPK elgsenos rinkimui

Šiame skyrelyje pateikiama svarbi procedūrų seka, kaip pilnai paruošti sistemą dinaminės KPK elgsenos rinkimui. Procedūrų seka taip pat pateikiama kartu su „Drak Management“ įrankiu. Norint tinkamai paruošti aplinką, svarbu griežtai vadovautis pateiktomis instrukcijomis.

Punktai, pažymėti žvaigždute (*) yra stipriai rekomenduojami, tačiau nėra privalomi. Jeigu šie punktai praleidžiami, didelė tikimybė, kad bus susidurta su problemomis. Taipogi rekomenduojama nekopijuoti komandų iš šio rašto darbo, bet perrašyti jas ranka.

1. Atidarius „terminal“, naudojant komandą „sudo su“, prisijungti su administratoriaus teisėmis. Visus tolimesniu veiksmus reikia atlikti su administratoriaus prieiga.
2. Įdiegti OS visus atnaujinimus.
3. Įdiegti „Python 3.8“

```
apt install software-properties-common
add-apt-repository ppa:deadsnakes/ppa
apt install python3.8
```

4. Įdiegti Redis serverį

```
apt install redis
```

5. Sudiegti „Drakvuf Sandbox“ įrankių paketus:

```
apt install ubuntu_focal_drakvuf-bundle-1.0-git20220222010225+fecea59-1-
generic.deb
apt install ubuntu_focal_drakrun_0.18.2_amd64.deb
apt install ubuntu_focal_drakcore_0.18.2_amd64.deb
```

6. Perkrauti sistemą, kad būtų paleista OS su „Xen“ hipervizoriumi

```
reboot
```

7. Pakartoti 1-ą punktą.
8. Sudiegti „Drakvuf Sandbox“ įrankio patobulinimus *:
    a. Iš „Drak Management“ įrankio pridedamo aplankalo „drakvuf-sb-fixes“ iškopijuoti ir įkelti visus failus į atitinkamas direktorijas – apkeisti failus.
    b. Atnaujinti „MiniO“ konfigūraciją, galima dviem būdais

i. Pirmasis būdas – iš „Drak Management“ įrankio pridedamo aplankalo „minio-configs“ iškopijuoti ir įkelti visus failus į atitinkamas direktorijas – apkeisti failus.

ii. Antrasis būdas – naviguoti į sistemos direkorija „/etc/drakcore/minio.env“ ir iškopijuoti „minio.env“ faile esančias „ MINIO_ACCESS_KEY“ ir „MINIO_SECRET_KEY“ reikšmės ir įklijuotas jas į „/etc/drakrun/config.ini“ ties eilute, kuri prasideda „access_key“ ir eilute, kuri prasideda „secret_key“.

c. Sudiegti trūkstamą „Python“ paketą „mslex“:

i. Įdiegti „pip“ (jeigu nėra įdiegtas)

```
apt install python3-pip
```

ii. Aktyvuoti „Python“ aplinką

```
source /opt/venvs/drakrun/bin/activate
```

iii. Įdiegti „mslex“

```
pip install mslex
```

9. Paruošti „Guest“ VM – įdiegti „Windows 7“

```
draksetup install '/abc/windows7.iso'
```

10. Kol vyksta „Guest“ VM pakrovimas, įdiegti „RealVNC Viewer“ – „Google“ paieškoje įvesti pavadinimą, parsiųsti „Linux“ „DEB x64“ paketą ir įdiegti

```
apt install paketo_pavadinimas.deb
```

11. „terminal“-e bus pateikiama „VNC“ prisijungimo nuoroda bei slaptažodis. Atidarius „VNC“ prisijungti ir atlikti „Windows 7“ diegimo procedūrą iki kol bus pasiektas OS darbalaukis. Tuomet reikia išjungti antivirusinę, jei tokia yra, ir perkonfigūruoti „user account control settings“ nustatymus pasirenkant „never notify“. Pavyzdys pridedamas šio darbo **Prieduose (5)**.

12. Jeigu norima vienu metu turėti daugiau nei 1 „Guest“ VM, atsižvelgiant į turimus sistemos resursus, galima tai padidinti su komanda „draksetup scale N“ N – skaičius.

13. „Guest“ VM diegimo pabaiga:

```
draksetup postinstall
```

Procesas apytiksliai gali užtrukti iki 2 val. 30min.

14. Programai baigus darbą, galima atlikti bandymą, atidarius naršykle, aplankomas puslapis „http://localhost:6300/“ ir jeigu puslapis pasikrauna, tuomet galima pamėginti įkelti „Windows“ vykdomąją programą sistemai ištestuoti.

15. „Drak Management“ įrankio diegimas:
    a. Iškelti „Drak Management“ aplankalą į norimą vietą, pavyzdžiui, „Documents“ direktoriją.
    b. Įdiegti „Python“ aplinkų konfigūravimo įrankį

```
apt install python3.8-venv
```

    c. Aktyvuoti „Python“ virtualią aplinką ir suteikti papildomų teisių esamam OS vartotojui

```
cd /opt/venvs
python3 -m venv drakmgmt
source drakmgmt/bin/activate
sudo chmod -R a+rwx drakmgmt („sudo“ jeigu iki šiol buvo vykdoma be
administratoriaus teisių)
```

    d. Naviguoti į iškelto „Drak Management“ aplankalą, sudiegti visus „Python“ paketus

```
pip install -r requirements.txt
```

16. „Drak Management“ įrankio aplankale atidaryti „config.ini“ su teksto redagavimo programa ir sukonfigūruoti įrankį pagal poreikius. Svarbu – atitinkamoje vietoje reikia įvesti OS vartotojo slaptažodį bei įvesti tokį pat „Guest“ VM kiekį, jeigu tai buvo konfigūruojama su komanda „draksetup scale N“.
17. Jeigu reikia išfiltruoti KPK, „config_sorting.ini“ faile pateikiama konfigūracija, o paleidimo komanda

```
python3 sort_samples.py
```

18. Paleisti „Drak Management“ įrankį su komanda

```
python3 app.py
```

Įvykdžius visus dokumentacijos žingsnius, tyrėjas gali vykdyti automatizuotus tyrimus su „Windows“ vykdomųjų failų pavyzdžiais. Kitame skyriuje atliekami empiriniai VMI-patobulinto metodo tyrimai.

5. Empiriniai VMI-patobulinto metodo tyrimai

Šiame skyriuje detaliai aprašomi visi dinaminės KPK elgsenos rinkimo patobulinimo, pritaikius VMI technikos paremtus įrankius, empirinio tyrimo etapai ir rezultatai. Pagrindiniai akcentai: tyrimui skirtų duomenų atrinkimas ir paruošimas, „Host“ bei „Guest“ aplinkų aprašymas, sukurto naujo įrankio „Drak Management“ konfigūracijos apžvalga, pralaidumo (tarpinių) tyrimų

analizė, galutinio tyrimo analizė, gautų po galutinio tyrimo rezultatų ir Čeponio tyrimo gautų rezultatų palyginimas, išvados.

5.1. Tyrimui skirtų duomenų atrinkimas

„Windows“ OS sisteminių kreipinių rezultatams rinkti buvo naudojamas naujausias tuo metu (2024m. 04mėn.) prieinamas KPK bandinių rinkinys – „Virus Share“ nr. 486. Šis rinkinys yra viešai ir nemokamai prieinamas parsisiųsti internete, „Virus Share“ bibliotekoje (pav. 5.1). Tačiau verta pabrėžti, jog prieiga prie KPK bibliotekos yra ribota ir norint gauti leidimą, būtina kreiptis į jos administraciją.

Detaliau apie bandinių rinkinį:

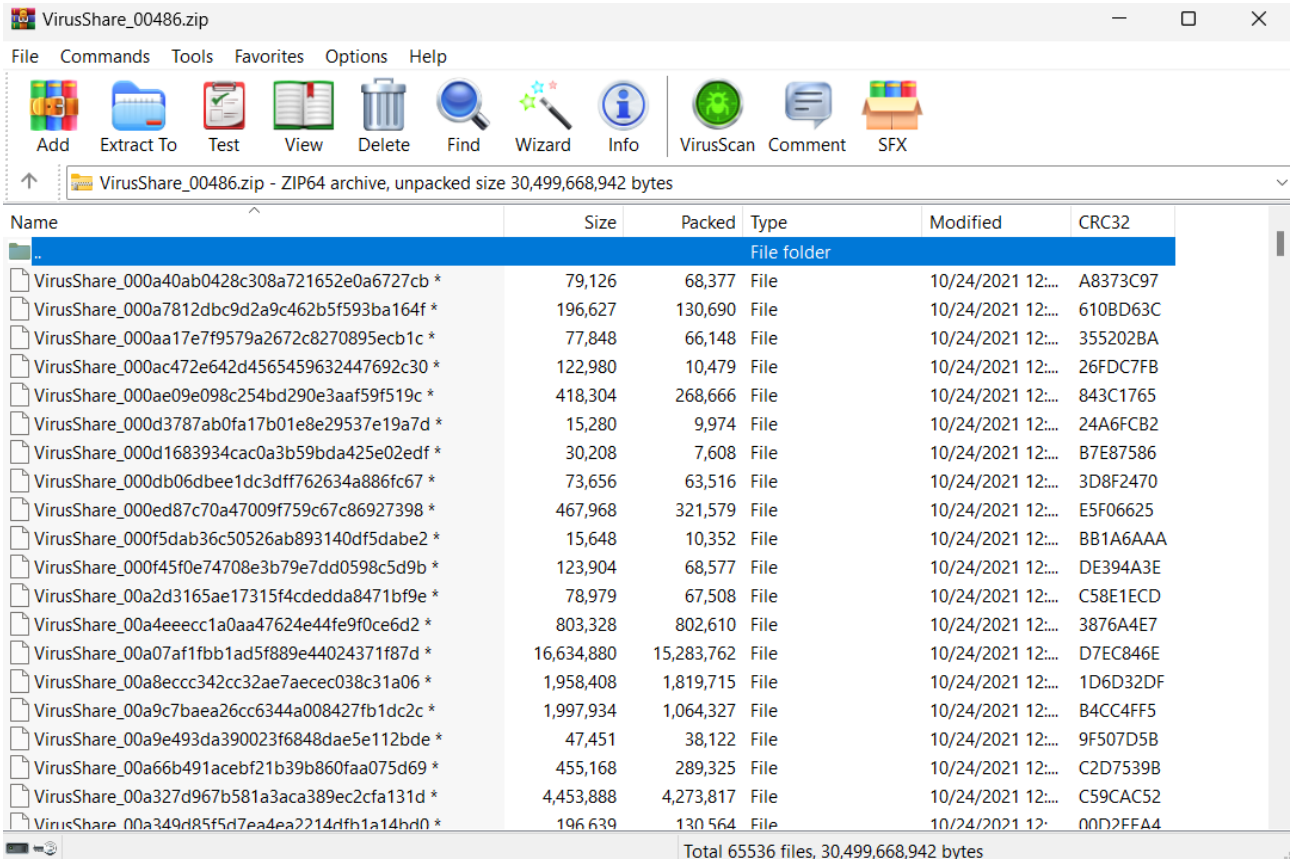
- Failo pavadinimas: „VirusShare_00486.zip“
- Failo „SHA256“ maiša: C096D42043ED1CA646A035A95FB02128E170F10B61A76118FE0BD708D3F87376
- Failo apimtis: 20.29GB
- Talpinimo bibliotekoje data ir laikas: 2024-02-14 16:29:00
- Nuoroda pateikiama literatūros šaltinių sąraše (VirusShare BitTorrent Tracker, n.d.).

VirusShare_00480.zip	19.56 GB	2023-09-09 13:41:21
VirusShare_00481.zip	24.11 GB	2023-09-22 13:51:57
VirusShare_00482.zip	28.97 GB	2023-10-21 13:04:27
VirusShare_00483.zip	36.96 GB	2023-11-04 12:10:06
VirusShare_00484.zip	24.15 GB	2023-11-10 09:16:07
VirusShare_00485.zip	46.76 GB	2024-02-07 17:37:12
VirusShare_00486.zip	20.29 GB	2024-02-14 16:29:00

5.1 pav. „Virus Share“ bibliotekoje patalpintas archyvas nr. 486, naudojamas šiame tyrime

5.2. Tyrimui skirtų duomenų paruošimas

Empirinio tyrimo etapams buvo pasirinktis bei išskaidytas „Virus Share“ nr. 486 KPK rinkinys, kurį galima matyti 5.2 iliustracijoje.



5.2 pav. „VirusShare“ KPK rinkinys

„Drak Management“ įrankio pagalba buvo atliktas KPK filtravimas, kad atskirti „Windows“ OS skirtas vykdomąsias programas nuo kitų OS. Pralaidumo tyrimas buvo atliekamas tris kartus. Kiekvieno karto metu KPK imtį sudarė 100 atsitiktinių programų pavyzdžių. Galutinis tyrimas buvo atliekamas vieną kartą, o jo KPK imtį sudarė 1000 atsitiktinių programų pavyzdžių.

„Drak Management“ konfigūracijos faile yra nurodomas „Windows“ vykdomųjų programų kiekis, kurį norima gauti atliekant KPK filtravimą. Filtravimo funkcionalumas aprašytas ankstesniame skyriuje. Galim išskirti tik tai, kad algoritmas veikia optimizuotai ir neapdoroja daugiau failų negu reikia, tad filtravimo procesas truko vos kelias sekundes. Galimai tam darė įtaką ir tai, kad tyrimui buvo naudojamas „SSD“ tipo kietasis diskas, kuris yra dešimtimis kartų spartesni už prieš dešimtmetį plačiau naudojamus „HDD“ tipo kietuosius diskus.

Mažiausiai saugyklos vietos užimantis „Windows“ vykdomasis pavyzdys siekė vos 2,6KB, o didžiausias – 22,4MB. Tai yra kone 86 tūkt. kartų didesnis failas. Iš viso 1000 pavyzdžių rezervuoja 1GB saugykloje. Kiti trys išfiltruoti KPK rinkiniai po 100 pavyzdžių vidutiniškai rezervavo apie 130MB. Verta pabrėžti, jog „Virus Share“ KPK rinkinio archyvinis failas yra suspaustas, tad išarchyvavus visus failus, bendra rinkinio svoris ženkliai padidėja.

5.3. Įrangos paruošimas

Dėl architektūros apribojimų privaloma naudoti specifinę techninę įrangą, kuri tenkintų šiuos reikalavimus: Intel procesorius, kuris turi palaiko „Intel VT-x“ ir „EPT“. Trumpai – tai „x64“ architektūros plėtiniai, leidžiantys paleisti virtualias mašinas tiesiogiai per procesorių.

Turint tokias galimybes, iš techninės pusės, visiems empirinio tyrimo etapams buvo naudojama aparatinė įranga, kurios esminiai komponentai yra:

- Operatyvioji atmintis – 16 GB „DDR4“
- Procesorius – „Intel Core i7 6700k“ (4 branduoliai)
- Pastovioji atmintis – naujas 512GB „Silicon Power A55“ SSD kietasis diskas („sata“ tipo).

5.3.1. *„Host“ aplinkos paruošimas*

Labai svarbu užtikrinti tinkamą komponentų sąveikavimą bei suderinamumą, todėl po daugelio bandymų buvo pasirinkta būtent ši, stabilumą užtikrinanti, konfigūracija:

- OS: „Ubuntu 20.04.6 LTS“ .
- Branduolio versija: „5.15.0-105-generic“.
- Suteikiama 8 GB operatyviosios (RAM) atminties.
- Suteikiamos 2 procesoriaus gijos (rekomenduojama suteikti mažiausiai 2 branduolius ir 4 gijas).
- „Redis“ naujausia versija.
- „Python 3.8“.
- „Drakvuf bundle 1.0“ („Xen“ hipervizorius 4.17 įeina į paketą).
- „Drakrun 0.18.2“ (dalis „Drakvuf Sandbox“).
- „Drakcore 0.18.2“ (dalis „Drakvuf Sandbox“).
- Modifikuotas „Drakvuf Sandbox“ kodas, užtikrinantis geresnį „drak postinstall“ komandos veikimą.
- „Drak Management“ – naujasis įrankis, pristatytas šiame darbe.

- ClamAV – atvirojo kodo nemokamas failų skanavimo įrankis, periodiniam „Host“ OS skanavimui, papildomam saugumui užtikrinti, nes dirbama su pavojingais KPK pavyzdžiais.

Įžvalgos, kiti pastebėjimai:

- „Ubuntu 18“ OS nesuderinama su įrankiais, nebuvo pasiektas norimas rezultatas. „Xen“ bei modifikuotas branduolys užkerta kelia OS, sistema užstringa begaliniam laikui, sistema negali pasiekti darbalaukio.
- „Ubuntu 22“ OS taip pat buvo atsisakyta dėl didėlių sistemos nestabilumų ir „kernel panic“ įdiegus „Xen“ ir „Drakvuf“ įrankiams.
- „Debian“ „Linux“ OS nebuvo testuojama.
- Naujesnės ar senesnės nei 5.15 branduolio versijos sukelia nestabilumo problemas, todėl nebuvo naudojamos. Atveju, kai buvo testuojama su „Ubuntu 22“, stabilumas pagerėjo grąžinus branduolio versiją į 5.15.

5.3.2. *„Guest“ aplinkos paruošimas*

Būtina atsižvelgti į aparatinės įrangos galimybes ir tyrimo tikslus bei rasti tinkamiausią sprendimą. Vienas iš reikalavimų yra užtikrinti kelių „Guest“ virtualių mašinų veikimą vienu metu, todėl operatyvioji atmintis buvo suskaidyta priskiriant 8 GB „Host“ mašinai ir po 3 GB dvejoms „Guest“ virtualioms mašinoms. Siekiant atlikti kuo geresne palyginamąją analizę, bus naudojamas „Windows 7 Pro x64“ OS atvaizdas. Šis atvaizdas pateikiamas literatūros šaltinių sąraše (Henry1679, 2020).

Plačiau apie „Windows 7“ atvaizdą 5.1 lentelėje:

5.1 lentelė. Windows 7 atvaizdo detalizavimas

Pavadinimas	„Windows 7 Professional SP1 (64 bit ISO)“
Publikacijos autorius	„Microsoft Corporation“
Patalpinimo data ir laikas	2020-06-16 20:41:53
Publikacijos data	2011
OS kalba	Anglų
ISO pavadinimas	GSP1RMCPRXFRER_EN_DVD.ISO
ISO apimtis	3.1 GB
SHA1 maiša	6e1cdc8612aa4872a3561c02dc2e5513ec9109ea

Verta pabrėžti, kad šio darbo rašymo metu, „Drakvuf“ bei „Drakvuf Sandbox“ autoriai rekomenduoja naudoti „Windows 7“ OS. Tai užtikrina didžiausią suderinamumą bei stabilumą. „Windows 10“ – galimas pasirinkimas, tačiau nerekomenduojamas dėl nestabilumo bei fakto, kad šiuo metu galima naudoti tik vieną iš pirmųjų (seniausių) „Windows 10“ versijų.

Tyrimo metu buvo atliekama daug bandymų su „Windows 7, „Windows 10“ ir šiuo metu naujausia – „Windows 11“ OS. Tačiau nepavyko pasiekti teigiamų rezultatų tiek su „Windows 10“, tiek su „Windows 11“ OS.

„Guest“ virtualią mašiną su „Windows 10“ OS pavyko paruošti tik iš dalies – pagrindinė „Drakvuf Sandbox“ „Python“ kalba parašyta rašmena labai prastai sukonfigūravo „Rekall“ profilius, o testavimo metu, KPK pavyzdžių šį aplinka visai nepriėmė. Tad po daugybės bandymų, buvo priimtas sprendimas nenaudoti „Windows 10“ tyrime.

„Guest“ virtualią mašiną su „Windows 11“ OS nepavyko paruošti. „Drakvuf Sandbox“ „Python“ kalba parašyta rašmena apskritai nesugebėjo parengti VM. Pagrindinė problema yra tai, kad vienos iš kritinių rašmenos funkcijų yra priklausoma nuo „Microsoft“ duomenų serverių. Šie serveriai teikia kritines „Windows 7“ ir „Windows 10“ OS komponentes „Rekall“ profiliams kurti, be kurių neįmanoma analizuoti. Deja, bet šiuo metu „Microsoft“ neteikia „Windows 11“ komponenčių, todėl analizė iš esmės yra neįmanoma.

5.4. Pirmasis empirinio tyrimo etapas

Šio tyrimo etapo tikslas yra nustatyti sąveikaujančių komponentų stabilumą bei pralaidumą. Tyrimas buvo atliekamas 3 kartus, KPK imtis: 3 grupės po 100 pavyzdžių, o vienu metu veikiančių „Guest“ VM kiekis: 2.

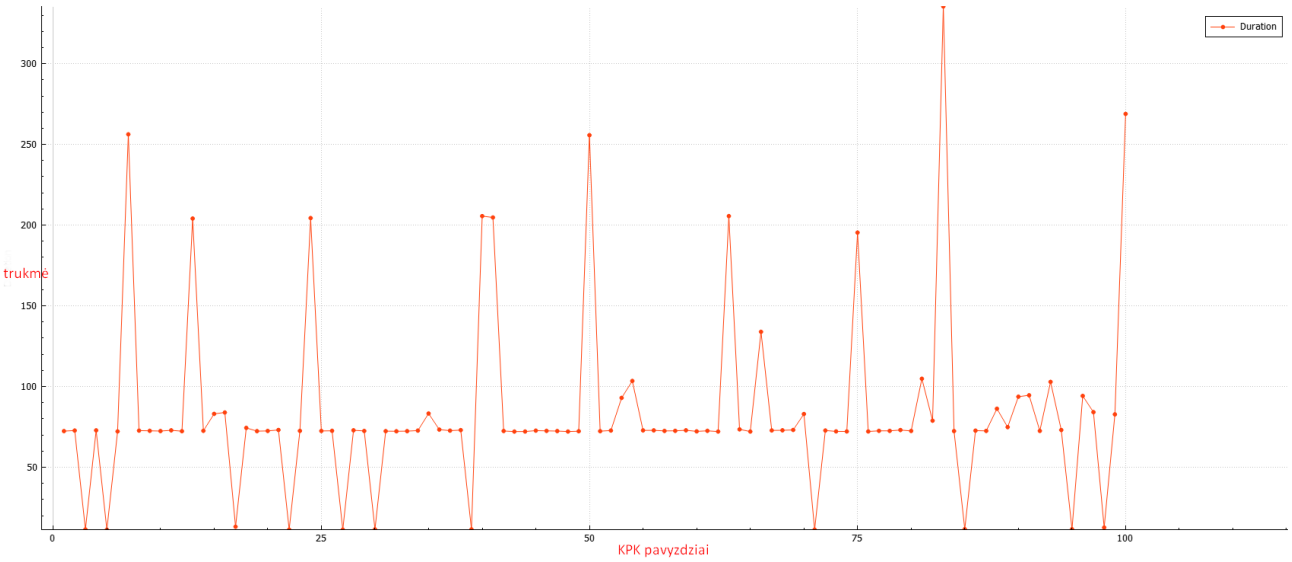
Atliktus 3 badymus buvo nustatyta, jog vidutinė eksperimento trukmė apytiksliai yra 1 valanda ir 30 minučių. Sėkmingai paleistų KPK pavyzdžių vidurkis – 77 iš 100. Taip yra dėl to, kad ne visos kenkėjiškos programos buvo sėkmingai paleistos. To priežastis yra nesuderinamumas tarp kenkėjiškos programos ir beveik 15 metų senumo „Windows 7“ OS. „Drakvuf“ „drakrun“ analizių žurnaluose buvo įžvelgta dažniausiai pasitaikanti nesėkmingo paleidimo priežastis – „ERROR_BAD_EXE_FORMAT“. Tai gali reikšti vieną ar daugiau dalykų – arba pats KPK pavyzdys yra nekokybiškas, arba tai yra architektūriniai nesuderinamumai tarp 64 bitų ir 32 bitų, arba KPK negali rasti tam tikro sisteminio komponento, pavyzdžiui bibliotekos, nuo kurios KPK yra priklausomas. Priežastis lieka ne iki galo atskleista, bet žinoma tai, kad KPK dėl nesuderinamumo su „Windows 7“, nebuvo paleistas analizei atlikti.

Rezultatų analizei atsitiktiniu būdu pasirinktas vienas iš trijų bandymų. **Taip pat šio darbo Prieduose (6) yra demonstracinė vaizdinė medžiaga, patalpinta „YouTube“ puslapyje**. Taip pat pateikiama literatūros sąraše (Švabauskas, 2024). Demonstracijoje vykdomas eksperimentas su 100 KPK pavyzdžių. Iš pradžių atliekamas KPK filtravimas, kad iš „Virus Share“ nr. 486 rinkinio atsitiktine tvarka išfiltruoti 100 vykdomųjų programų, skirtų „Windows“ OS. Tuomet atliekamas

minimalus „Drak Management“ įrankio konfigūravimas ir galiausiai tyrimo vykdymas. Video medžiaga išgauta pasitelktus išorinius įrenginius, tiesiogiai prijungtus prie „Host“ mašinos vaizdo išvesties „HDMI“ kabelio. Būtent toks fiksavimo būdas užtikrina tai, kad nebus daroma įtaka „Host“ mašinos našumui. Video demonstracijoje kartais iškreipiamas vaizdas, tai yra ne „Host“ mašinos, o išorinių įrenginių kaltė. Toliau atliekama šio eksperimento analizė, duomenys pateikiami grafikų iliustracijose bei lentelėse.

Grafike (pav. 5.3) yra vaizduojamos KPK pavyzdžių tyrimų trukmės.

- Horizontali ašis – išrikiuoti KPK pavyzdžiai nuo 1 iki 1000.
- Vertikali ašis – analizės trukmė (sekundės).



5.3 pav. KPK pavyzdžių tyrimų trukmės

Svarbu atkreipti dėmesį į kelis aspektus:

- 1 KPK pavyzdžiui išanalizuoti skiriamos 60 sekundžių.
- Po 60 sekundžių tyrimas stabdomas.
- Laikas „Guest“ virtualioje mašinoje nėra vientisas. Tai reiškia, jog priklausomai, nuo aparatinės įrangos apkrovos tam tikru laiko metu ir bendro našumo, „Guest“ VM gali funkcionuoti lėčiau, todėl laiko tėkmė yra dinamiška.

Todėl šiame grafike galima įžvelgti, kad kiekvieno KPK analizės trukmė variuoja tarp labai trumpos (11-13 sekundžių, kuomet atkuriama „Guest“ VM pradinė būsena bei įkeliamas KPK), vidutinės (normalios) ir ilgos (kuomet „Guest“ VM veikė daug kartų lėčiau, nei tikėtasi). Daugiau įžvalgų pateikiama 5.2 lentelėje:

5.2 lentelė. Greitaveikos ir pralaidumo KPK pavyzdžių tyrimo statistinė lentelė

Vidutinė vieno KPK analizės trukmė (visi KPK pavyzdžiai)	85.23 sek.
Vidutinė vieno KPK analizės trukmė (tik sėkmingai ištirti KPK pavyzdžiai)	94.49 sek.
Trumpiausiai trunkanti vieno KPK analizė	11.28 sek.
Ilgiausiai trunkanti vieno KPK analizė	335.44 sek.
Bendra viso tyrimo trukmė	1 val. 23 min. 35 sek. arba 5015.48 sek.
Bendra viso tyrimo trukmė, jei būtų naudojama tik viena „Guest“ VM konfigūracija (apskaičiuot susumavus visų KPK pavyzdžių analizių trukmes)	2 val. 22min. 03 sek. arba 8523 sek.

Tyrimas, kaip jau buvo minėta, buvo vykdomas su dviejų vienu metu veikiančių „Guest“ VM konfigūracija. Galima daryti prielaidą, kad tyrimo trukmė gali padidėti 1,7 karto, jei būtų naudojama konfigūracija, kurioje vienu metu veikiančių „Guest“ VM kiekis būtų lygus 1. Tokiu atveju, tyrimas truktų 2 val. 22min. 03 sek., arba 8523 sek.

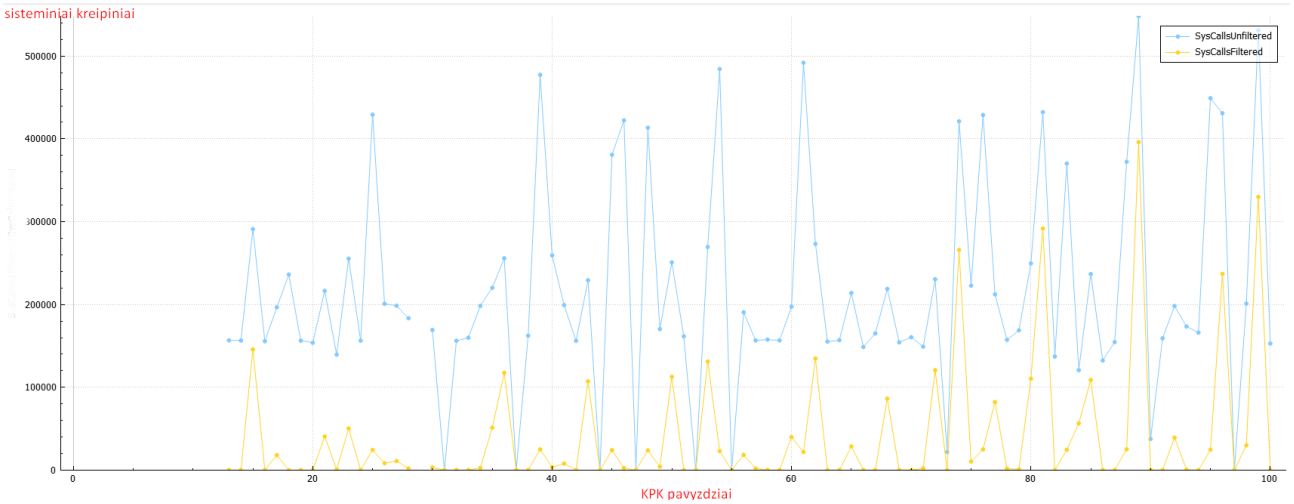
Galima palyginti pralaidumą su Čeponio vykdyto tyrimo rezultatais, kuomet 12110 KPK pavyzdžių analizė truko du mėnesius (2017.07-2017.08) naudojant 6 tuo pat metu veikiančių „Guest“ VM konfigūraciją. Tai apytiksliai sudaro 62 nepertraukiamos KPK analizės dienas. 62 paros yra lygios 1488 valandoms. 1488 valandos yra lygios 89280 minutėms. 89280 minutės yra lygios 5356800 sekundėms.

Remiantis tokiais preliminariais paskaičiavimais, galima daryti išvadą, jog 1 KPK išanalizuoti reikėjo apytiksliai 442 sekundžių arba 7 min. 22 sek. O tai yra 5,2 karto lėčiau palyginus su šiuo pralaidumo tyrimu. Tačiau į šio palyginimo rezultatus vertėtų žiūrėti labai nuolankiai, kadangi tiek tyrimų aplinkos, tiek konfigūracija, tiek KPK rinkiniai bei architektūros yra labai skirtingos.

Jeigu šiame darbe būtų naudojama galingesnė aparatinė įranga ir būtų pavykę suvienodinti vienu metu veikiančių „Guest“ VM kiekį iki 6, tuomet pralaidumas dar labiau padidėtų. Tačiau jau dabartinės statistikos pakanka teigti, kad šis dinaminės KPK elgsenos rinkimo metodas yra efektyvesnis palyginus su Čeponio sukurtu metodu.

Grafike (pav. 5.4) vaizduojami nefiltruotų ir filtruotų sisteminių kreipinių dažniai.

- Horizontali ašis – išrikiuoti KPK pavyzdžiai nuo 1 iki 1000.
- Vertikali ašis – sisteminių kreipinių, kuriuos pavyko užfiksuoti, kiekis.
- Mėlyna linija ir taškai pažymi nefiltruotus sisteminių kreipinių kiekius.
- Geltona linija ir taškai pažymi filtruotus sisteminių kreipinių kiekius.



5.4 pav. Nefiltruotų ir filtruotų sisteminių kreipinių dažniai

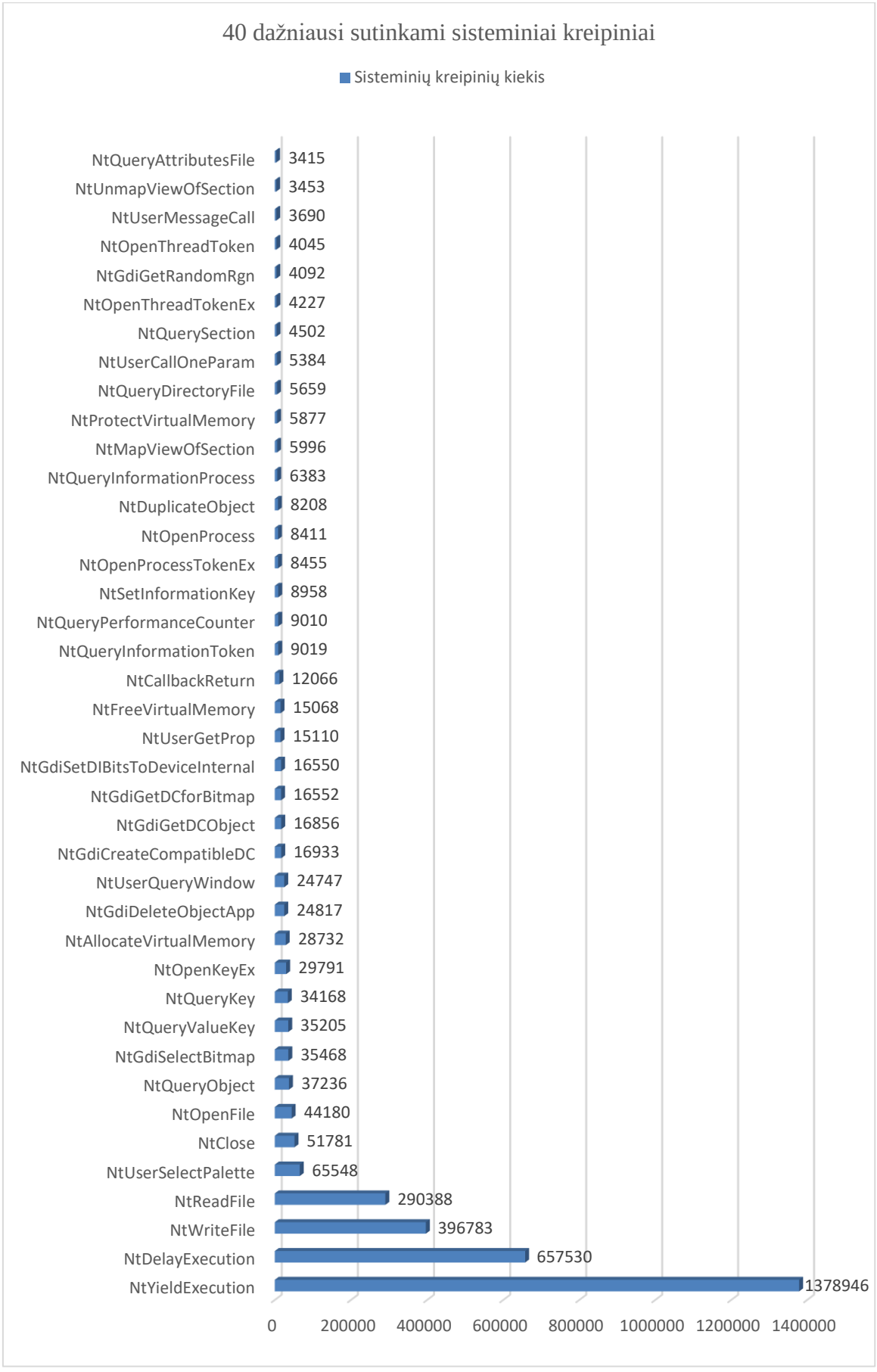
Greitaveikos tyrimo metu gauti sisteminių kreipinių kiekių rezultatai, kurie parodo tiek filtruotus pagal KPK iššauktus procesų sisteminius kreipinius, tiek nefiltruotus pagal KPK sisteminius kreipinius. 5.3 lentelėje pateikti sisteminių kreipinių kiekiai rodo, kad filtruoti sisteminiai kreipiniai sudaro 18 % visų sisteminių kreipinių, kurie buvo iššaukiami įvairių sisteminių procesų. Būtent filtravimas leido atrinkti tik svarbius analizei sisteminius kreipinius, kurie turi tiesioginį ryšį su KPK.

5.3 lentelė. Greitaveikos ir pralaidumo tyrimo metu surinktų sisteminių kreipinių statistinė lentelė

Nefiltruotų sisteminių kreipinių kiekis	18 742 755
Filtruotų sisteminių kreipinių kiekis	3 461 609
Sisteminių kreipinių duomenų bazės dydis	1.3 GB
Bendra viso tyrimo trukmė	1 val. 23 min. 35 sek. arba 5015.48 sek.

Kaip buvo aptariama ankstesniame skyriuje, filtravimo algoritmas atsižvelgia į du dalykus: proceso identifikacinį numerį bei tėvinį identifikacinį numerį – „PPID“. Tai padeda išfiltruoti sisteminius kreipinius, kurie buvo iššaukti paties KPK proceso bei jo dinamiškai kuriamų dukterinių procesų.

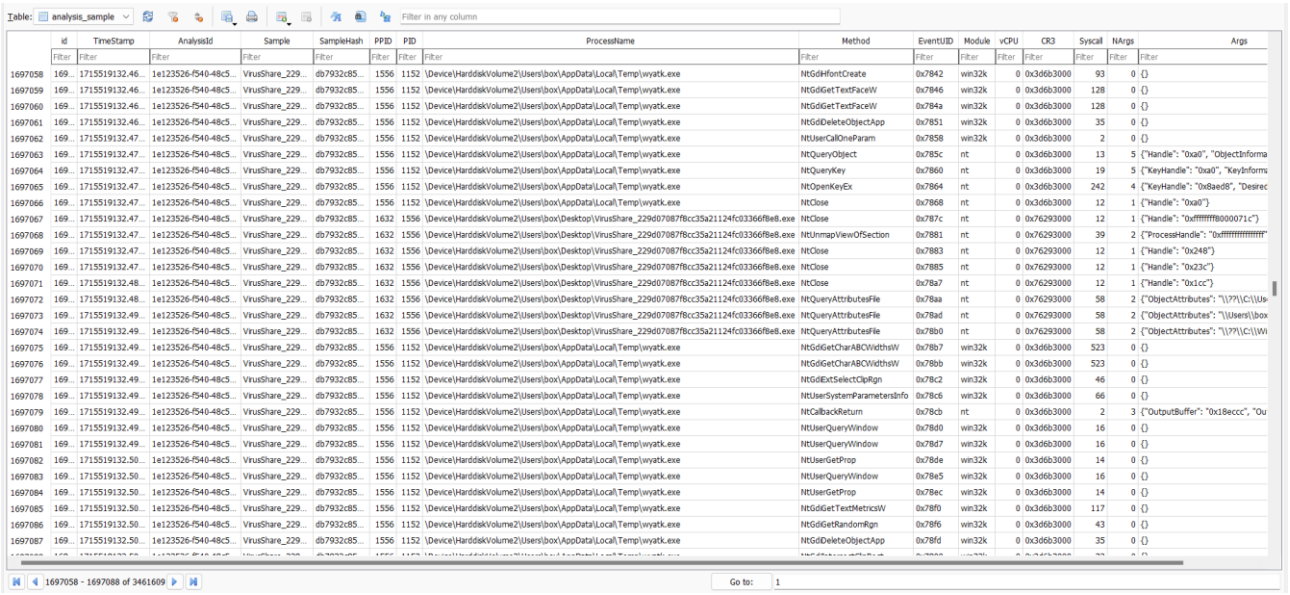
Iš viso buvo surinkti 326 unikalūs sisteminiai kreipiniai, kurie visų bendras iškviečiamų kiekis yra 3461609 kartai. 40 dažniausiai iškviečiamų sisteminių kreipinių sąrašas pateikiamas grafike (pav. 5.5). Joje horizontalią ašį atspindi sisteminių kreipinių kiekis, o vertikalioje ašyje išvardinti sisteminiai kreipiniai.



5.5 pav. 40 daugiausiai surinktų sisteminių kreipinių dažnumas

Labiausiai išskirtinas sisteminis kreipinys – „NtYieldExecution“ su 1,3 mln. iškvietimų. Toliau seka „NtDelayExecution“, „NtWriteFile“, „NtReadFile“ bei „NtUserSelectPalette“.

Sisteminių kreipinių duomenų bazėje pavyzdys pateikiamas 5.6 iliustracijoje. Joje matyti šio tyrimo atsitiktiniai sisteminių kreipinių žurnalizuoti įrašai. Tai yra puikus sisteminių kreipinių filtravimo pavyzdys, kuomet yra matomas tėvinis procesas „\Device\HarddiskVolume2\Users\box\Desktop\VirusShare_229d07087f8cc35a21124fc03366f8e8.exe“, kurio identifikacinis numeris „PID“ yra 1556 bei vienas iš dukterinių procesų – „\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\wyatk.exe“.



5.6 pav. Sisteminių kreipinių duomenų bazėje pavyzdys

Reikėtų pabrėžti, kad „Drakvuf“ surenkamų sisteminių kreipinių įrašai saugo daugiau informacijos, tačiau šiame dinaminės KPK elgsenos pagerinimo metode buvo panaudota ir atrinkta labiausiai svarbi informacija, siekiant optimizuoti duomenų bazės dydį neperkraunant jos mažiau reikšmingais dalykais.

5.5. Galutinis empirinio tyrimo etapas

Šio tyrimo etapo tikslas yra nustatyti ilgalaikį dinaminės KPK elgsenos rinkimo patobulinimo veiksmingumą bei stabilumą. Tyrimas buvo atliekamas 1 kartą, KPK imtis: 1000 pavyzdžių, o vienu metu veikiančių „Guest“ VM kiekis: 2. Šio tyrimo rezultatų failai pateikiami šio darbo literatūros sąraše (Švabauskas, 2024), o informacija apie failus – **Prieduose (7)**.

Gulinio empirinio tyrimo etape atliktas eksperimentas truko 37 val. 15min. 22 sek. Sėkmingai paleistų KPK pavyzdžių kiekis – 727 iš 946, tai yra 76 % sėkmės rodiklis. Taip yra dėl to, kad ne visos kenkėjiškos programos buvo sėkmingai paleistos. Kaip buvo minima ankstesniame skyriuje,

didžiausias veiksnys yra kenkėjiškos programos ir „Windows 7“ OS nesuderinamumas. Tačiau šio tyrimo etape, kiek kitaip nei praeitame etape, ne visi išanalizuoti KPK duomenys buvo surinkti bei agreguoti.

Eksperimento metu, po apytiksliai 400 KPK paleistų pavyzdžių, „Host“ sistemos našumo rodiklis palaipsniui krito. Prieš eksperimentą jau buvo žinoma, jog tyrimui naudojama konkreti aparatinė įranga yra nepakankamai galinga vienu metu ilgesniam laikotarpiui atlikti dviejų „Guest“ VM darbą. Taip yra dėl to, kad naudojamas minėtas „Intel i7 6700k“ procesorius, kuris turi 4 branduolius ir 8 gijas. „Drakvuf“ konfigūracija leidžia tyrėjui keisti aparatinės įrangos resursų padalijimą, tačiau nepaisant to, procesoriaus našumas bei jo branduolių kiekis stipriai apriboja tyrimo vykdymą.

D. Čeponio eksperimente, kaip jau buvo minėta šio darbo viename iš skyrių, buvo naudojama aparatinė įranga, kuri labiau skirta darbui su virtualiomis mašinomis. Naudojamoje įrangoje sumontuoti du „Intel Xeon“ procesoriai su 4 branduoliais suteikė dvigubai didesnį kiekį branduolių. Dėl to D. Čeponio tyrimas buvo vykdomas su 6 vienu metu veikiančiomis VM. Šio darbo atveju, buvo naudojama tai, kas turima, todėl tam, kad įrodyti, jog dinaminės KPK elgsenos rinkimo metodas yra pagerintas, buvo siekiama pademonstruoti dviejų vienu metu veikiančių „Guest“ VM funkcionalumą ilguoju laikotarpiu.

Taip pat reikia paminėti, kad D. Čeponio tyrime buvo naudojami daugiau nei keli kietieji diskai „Raid 0“ konfigūracijoje, kas leido sujungtus N-kiekį kietųjų diskų naudoti kaip vieną vienetą, ko pasekoje buvo padidintas sistemos našumas bei stabilumas. Šio tyrimo atveju, buvo naudojamas vienas kietasis diskas 512 GB talpos. Po daugiau nei paros intensyvių skaitymo ir rašymo operacijų, kietojo disko darbinė temperatūra buvo ganėtinai aukšta. Sistema pradėjo ganėtinai lėtai veikti. Pačios operacijos su failais truko daug kartų ilgiau, nei įprastai.

Reikia paminėti, kad „Drak Management“ tiesiogiai priklauso nuo kitų sisteminių komponentų bei tokių įrankių, kaip „Drakvuf“ bei „Drakvuf Sandbox“. Sulėtėjus sistemos veiksmingumui, sulėtėjo ir kiti kritiniai duomenų gavimo bei apdorojimo komponentai. „Drakvuf“ įrankio duomenų apdorojimas truko ilgiau, „Drakvuf Sandbox“ API grąžinamų rezultatų užklausų trukmė ilgėjo.

Kuriant „Drak Management“ įrankį, buvo atsižvelgiama į šias rizikas, todėl labai gerai pasitarnavo mechanizmas, kuris atlieka „request“ užklausų pakartojimą gavus klaidos pranešimą iš „Drakvuf Sandbox“ API.

Galutinio tyrimo etapo trukmė ženkliai didesnė nei buvo prognozuojama (15 – 20 valandų). Tai yra kone dvigubai ilgiau, nei buvo numatyta. Tačiau tai leido suprasti, kad norint atlikti tyrimą su

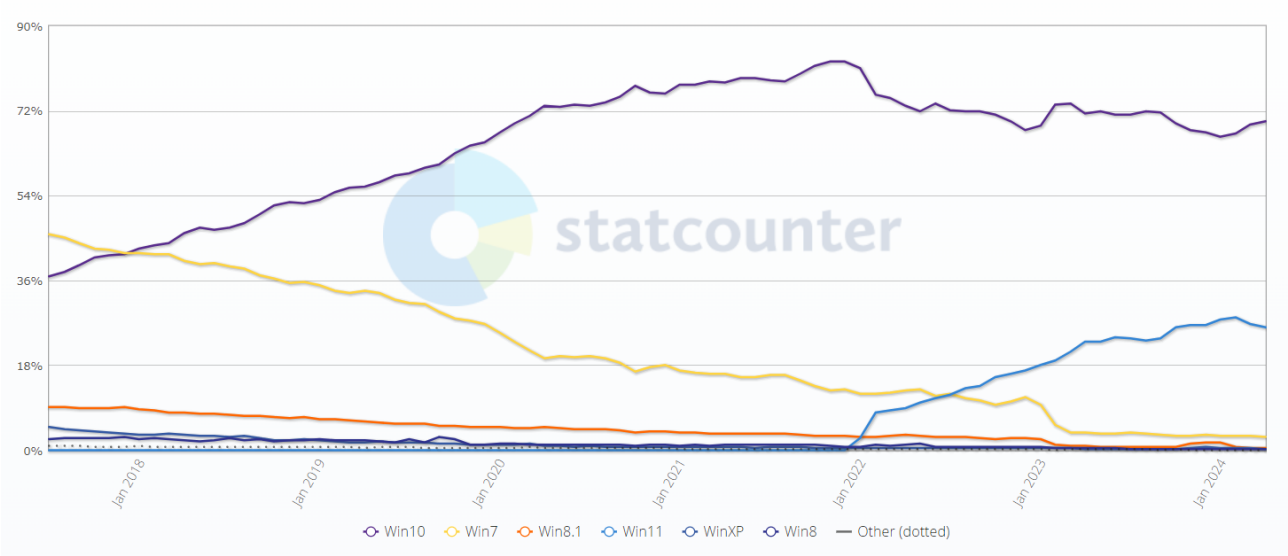
dideliu kiekių KPK, reikia ir galingesnės įrangos. Bet tai nedraudžia keisti konfigūraciją ir naudoti tik vieną „Guest“ VM arba sumažinti KPK kiekį iki 200-300 imties ir atlikti eksperimentus daug kartų. Kitą vertus, sumažinti KPK imtį nėra praktiška jeigu norima ištestuoti labai didelį kiekį pavyzdžių. Todėl rekomenduojama atsižvelgti į šio tyrimo rezultatus ir pasirinkti geriausiai šiai užduočiai vykdyti aparatinę įrangą. Taigi siekiant geriau suprasti tyrimo galutinį etapą, toliau apžvelgiami gauti empirinio tyrimo rezultatai 5.4 lentelėje.

5.4 lentelė. Galutinio empirinio tyrimo etapo suvestinė lentelė

Bendra viso tyrimo trukmė	37 val. 15 min. 22 sek. arba 134121.61 sek.
Pradžios data	2024 05 11
Pabaigos data	2024 05 12
KPK imtis	1000 pavyzdžių
Sėkmingai paleisti pavyzdžiai	727
Nesėkmingai paleisti pavyzdžiai	219
Pavyzdžių kiekis, iš kurių buvo sėkmingai surinkti duomenys	946
Sėkmės rodiklis	76.849 %
Sisteminių kreipinių duomenų bazės dydis	8.5 GB
Sesijos duomenų bazės dydis	260 KB
Agreguotų sisteminių kreipinių duomenų bazės dydis	20 KB
Komandinės eilutės išvesties kopijos dydis	4.69 MB
„Drakrun“ servisų kiekis („Guest“ VM kiekis)	2
„Drak Post Process“ servisų kiekis	1

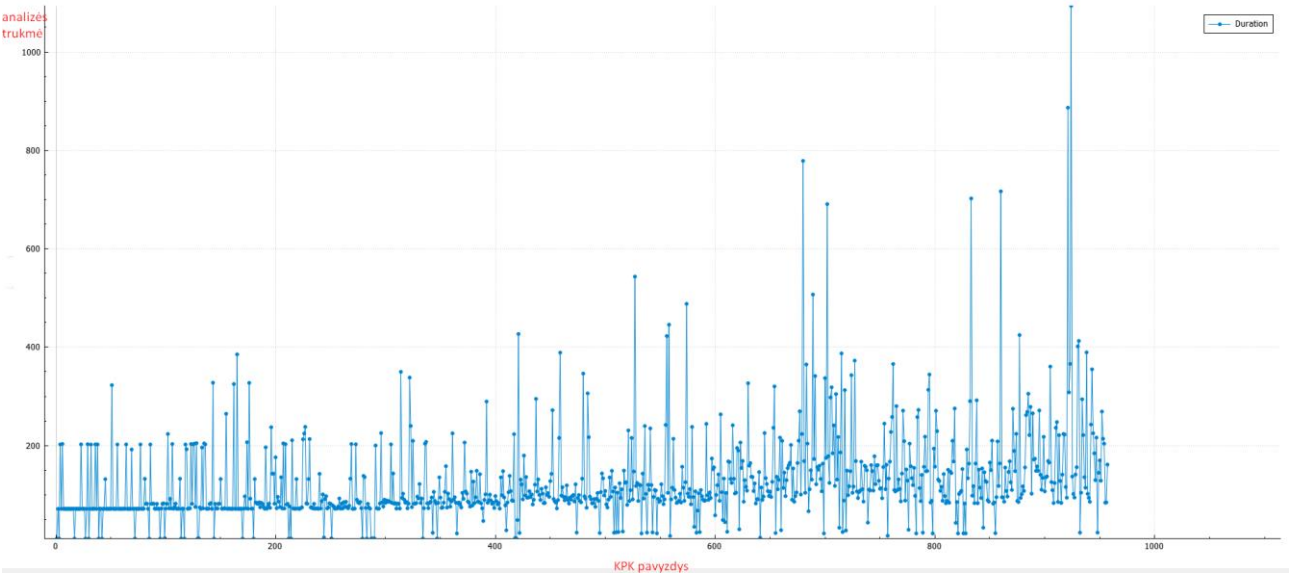
Analizės sėkmės rodiklis – 76,84 %. Tai rodo beveik 8 iš 10 KPK buvo sėkmingai paleisti. Visgi naudojamas pats naujausias, šio darbo atlikimo metu, KPK rinkinys, tad nenuostabu, jog ne visus KPK pavyzdžius pavyko sėkmingai paleisti „Guest“ aplinkoje, kurioje įdiegta „Windows 7“ OS. Dėl anksčiau minėtų apribotos aparatinės įrangos priežasčių, „Drak Management“ surinko 946 KPK analizių įrašus.

Galima daryti prielaidą, kad sėkmės rodiklis galėtų būti aukštesnis, jeigu būtų naudojamas ekvivalentus KPK rinkinys, koks buvo naudojamas D. Čeponio tyrime. Verta pabrėžti, kad laiko tarpas taip D. Čeponio sudaryto rinkinio ir šiame tyrime naudojamo „Virus Share“ nr. 486 rinkinio – beveik 7 metai. Šio laikotarpio pradžioje „Windows 7“ OS dominavo OS rinką (pav. 5.7). Tuo metu „Windows 7“ dominavo rinką su 45,73 %, o antroje vietoje puikavosi „Windows 10“ su 36,93 %. Tačiau jau 2018 metų pradžioje, „Windows 10“ nukarūnavo „Windows 7“, ir nuo tada įžvelgiamas stabilus „Windows 7“ kritimas. Ypatingai didelį šuolį žemyn galima matyti 2023 metų pradžioje. Šiai dienai, 2024 metų balandžio mėnesį, remiantis pateiktos statistikos duomenimis, „Windows 7“ OS naudojasi 2,81 %. Tai yra 16 kartų mažiau, nei prieš beveik 7 metus. Todėl galima teigti, kad naujausio KPK rinkinio kenkėjiškos programos yra labiausiai suderinamos su „Windows 10“ OS, antroje vietoje su „Windows 11“ OS.



5.7 pav. Windows OS rinkos pasiskirstymo grafikas nuo 2017-07 iki 2024-04 (StatCounter, 2024)

Anksčiau buvo minėta, kad eksperimento metu, po apytiksliai 400 KPK paleistų pavyzdžių, „Host“ sistemos našumo rodiklis palaipsniui krito. Tai galima matyti galutinio etapo KPK pavyzdžių tyrimų trukmių grafike (pav. 5.8).



5.8 pav. Galutinio etapo KPK pavyzdžių tyrimų trukmės

Kad geriau suprasti KPK analizių trukmes, pateikiama statistinė lentelė 5.5, kurioje nurodomos vidutinės trukmės, trumpiausia bei ilgiausiai trunkanti KPK analizė, bendra tyrimo trukmė (nuo pirmo KPK pavyzdžio analizės pradžios iki paskutinio KPK pavyzdžio analizės pabaigos).

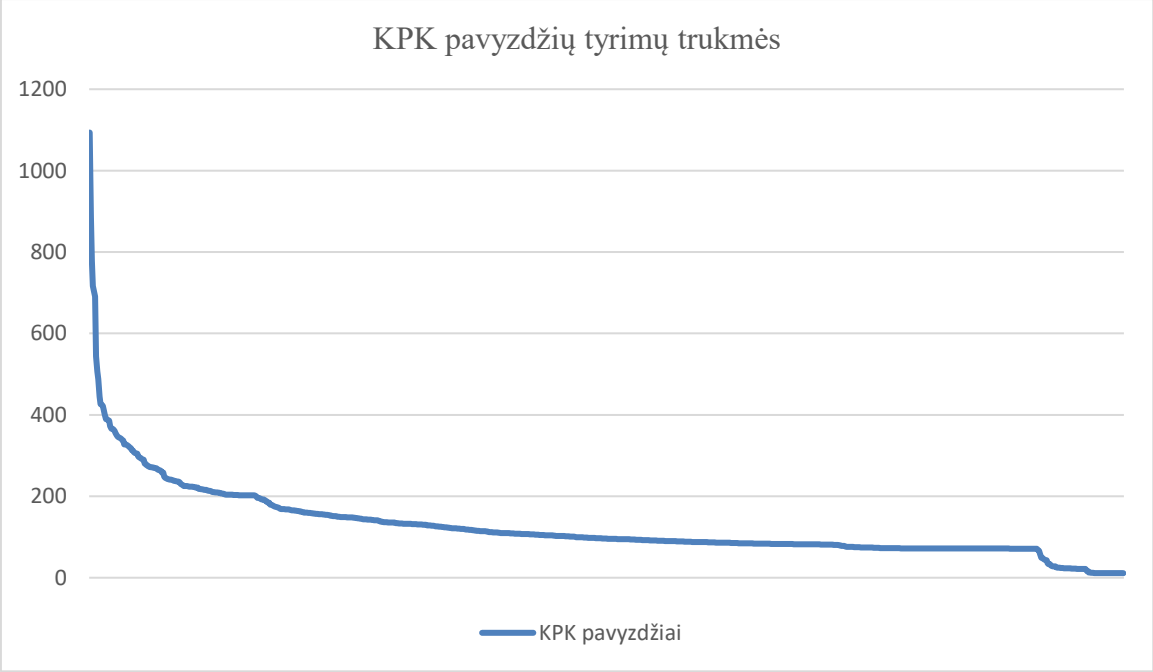
5.5 lentelė. Galutinio etapo greitaveikos ir pralaidumo KPK pavyzdžių tyrimo statistinė lentelė

Vidutinė vieno KPK analizės trukmė (visi KPK pavyzdžiai)	125.75 sek.
Vidutinė vieno KPK analizės trukmė (tik sėkmingai ištirti KPK pavyzdžiai)	130.80 sek.
Trumpiausiai trunkanti vieno KPK analizė	11.10 sek.
Ilgiausiai trunkanti vieno KPK analizė	1093.55 sek.
Bendra viso tyrimo trukmė (neįtraukiant „Drak Management“ duomenų kaupimo, apdorojimo ir saugojimo operacijų)	33 val. 25 min. 48 sek. arba 120348.66 sek.

Norima akcentuoti, kad didžioji dauguma KPK pavyzdžių analizių buvo atlikta beveik realiuoju laiko tempu. Kiek mažiau nei ketvirtadalis analizių prailgėjo dėl šiame tyrime naudojamos aparatinės įrangos perkrovos.

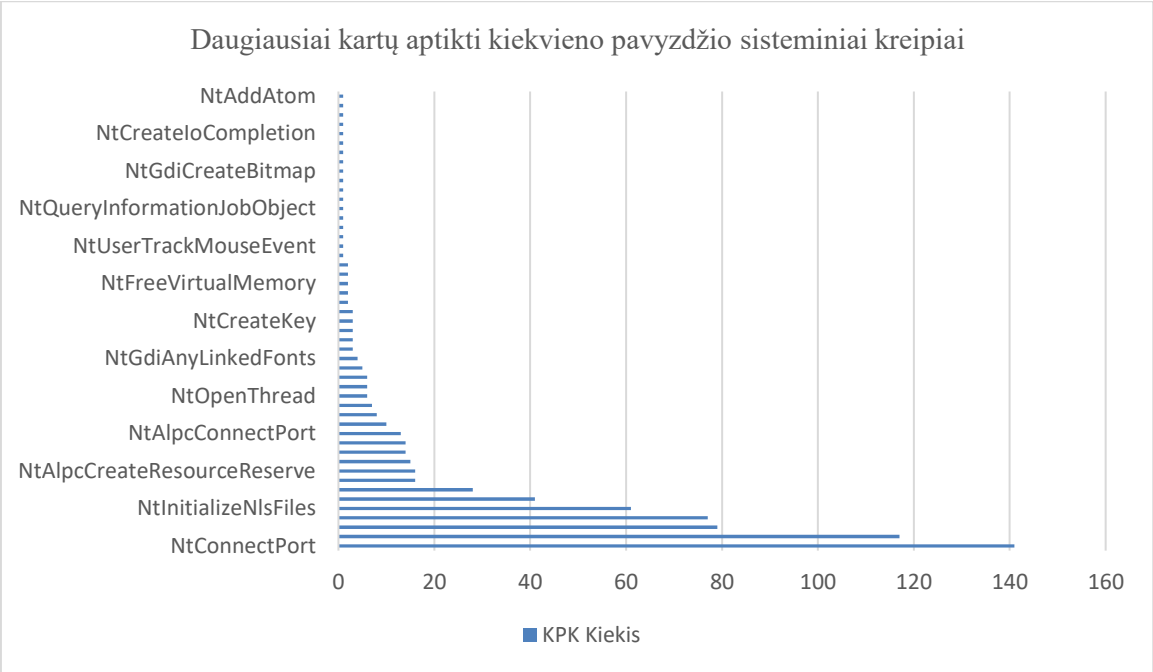
Grafike (pav. 5.9) vaizduojamos KPK pavyzdžių analizių trukmės.

- Horizontali ašis – išrikiuoti 1000 KPK pavyzdžiai nuo ilgiausios analizės iki trumpiausios analizės.
- Vertikali ašis – atvaizduojama KPK pavyzdžio tyrimo trukmė (sekundės).



5.9 pav. Galutinio etapo KPK pavyzdžių tyrimų trukmės nuo ilgiausios iki trumpiausios

Daugiausiai kartų aptiktų kiekvieno KPK pavyzdžio analizės sisteminių kreipinių grafiko iliustracija (pav 5.10) pateikiama žemiau. Joje pateikiami duomenys iš „Drak Management“ sukurtos sesijos duomenų bazės, kurioje pateikiami KPK analizių rezultatai.



5.10 pav. Galutinio etapo daugiausiai kartų aptikti KPK pavyzdžio sisteminiai kreipiniai

Kiekvienas įrašas sesijos duomenų bazėje talpina svarbiausius KPK analizės parametrus: analizės identifikacinį numerį, KPK pavyzdžio pavadinimą, KPK pavyzdžio „SHA256“ maišą, pradžios bei pabaigos datas, analizės trukmę, bandymo numerį, statusą, nefiltruotų ir filtruotų sisteminių kreipinių kiekius, daugiausiai ir mažiausiai kartų pasitaikančio sisteminio kreipinio pavadinimą (pav. 5.11).

	id	AnalysisId	SampleName	SampleHash	StartDate	EndDate	Duration	Attempt	Status	SysCallsUnfiltered	SysCallsFiltered	SysCallMostCommon	SysCallRarest
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
722	722	029e95f4-7ee2-466c-8218-d27c2fa04348	VirusShare_b2e64b7720e244e96c171388741e44b5	5ff58a6b6f6329d10...	Sat May 11 20:03:51 2024	Sat May 11 20:05:48 2024	117.3200776577	1	Pass	144079	164	NtConnectPort	NtProtectVirtualMemory
723	723	8a5b399d-d663-4262-9cfb-adae50d13c9c	VirusShare_dde15868037fcb7c84131cddc835076a	e22a0832f2ac39f59...	Sat May 11 20:03:51 2024	Sat May 11 20:06:19 2024	147.886249303818	1	Pass	141182	163	NtConnectPort	NtProtectVirtualMemory
724	724	33090f1b-b199-42ca-bf89-e6446b9ee3c6	VirusShare_7931a5d0263102aab173687926e6061e	812c3d1515133e65...	Sat May 11 20:13:32 2024	Sat May 11 20:19:15 2024	343.116004943848	1	Pass	142434	316	NtCreateEvent	NtClose
725	725	5ee44fe1-5f59-4852-8983-2058ef51214d	VirusShare_dce85886daad6eb87c3c67b5e071d641	f5518cbefbb78c40d...	Sat May 11 20:17:34 2024	Sat May 11 20:19:18 2024	104.035024166107	1	Pass	349081	13545	NtGdiCreatePatternBrushInternal	NtClose
726	726	d90747c8-61e1-49c0-a247-1d5132f65785	VirusShare_ef3b561c39cfdfaf450726466f9b558e	19acd05215ef3f637...	Sat May 11 20:31:55 2024	Sat May 11 20:33:52 2024	116.701310873032	1	Pass	142027	163	NtConnectPort	NtProtectVirtualMemory
727	727	36b7e729-1aad-4821-9c09-76a4ed767ea2	VirusShare_367277a8e5bfbeac6bb5d7c81567d34d	964eb87c494ec16b...	Sat May 11 20:27:39 2024	Sat May 11 20:33:52 2024	372.67956662178	1	Pass	134828	163	NtConnectPort	NtProtectVirtualMemory
728	728	9d165d51-e92a-4c31-95b2-3b4b549f3a5b	VirusShare_787cc165c68459ec64312260bee3bc0d	a3121984cecddff33...	Sat May 11 20:38:26 2024	Sat May 11 20:41:14 2024	168.560182571411	1	Pass	226587	163037	NtClose	NtGdiGetBitmapBits
729	729	beecc1ff-e4e7-4acd-a976-dad48cde3555	VirusShare_85066478794070b0b5f9f80e321e3f221	b942246ae6a107be...	Sat May 11 20:42:51 2024	Sat May 11 20:44:38 2024	107.00080537796	1	Pass	234543	92026	NtAccessCheck	NtDelayExecution
730	730	0fdb167e-0b7d-42ab-bebb-0d56e4db184d	VirusShare_80996d4881522b21a106bccec7e6af50	6b4acc0171d57fa95...	Sat May 11 20:45:26 2024	Sat May 11 20:46:59 2024	93.3715283870697	1	Pass	147828	163	NtConnectPort	NtProtectVirtualMemory
731	731	18b785f6-dfbe-4b2c-a5ac-5b9b2b948288	VirusShare_f2fb41f4581916dc7d87d65464b5e54b	65229ccd18fc71f9c...	Sat May 11 20:47:40 2024	Sat May 11 20:49:25 2024	105.588064670563	1	Pass	158679	330	NtConnectPort	NtQueryInformationProcess
732	732	224abc17-326d-4b27-b787-deb9a6390d0f	VirusShare_02137704a058604a65e67aa48a3e0e24	bc2d94a40c4e6fce2...	Sat May 11 20:49:10 2024	Sat May 11 20:50:58 2024	108.486466407776	1	Pass	151402	159	NtConnectPort	NtProtectVirtualMemory
733	733	65d18981-dc5e-48d5-9219-ab0c8492b40e	VirusShare_0fc347d70758a0477618ecde63bb4680	fff5e108c8ec8c342...	Sat May 11 20:51:50 2024	Sat May 11 20:54:38 2024	167.33021402359	1	Pass	148799	42	NtConnectPort	NtQueryAttributesFile
734	734	7b6533fa-bf42-4f88-b8ca-483f70f53b95	VirusShare_be7fd0df80d9555343de3a1303fcfc9d	35d5b8175281a353...	Sat May 11 20:59:58 2024	Sat May 11 21:01:50 2024	111.437035322189	1	Pass	242920	74440	NtAlpcSendWaitReceivePort	NtUserQueryWindow
735	735	5806653b-3dca-4d88-8b8e-b6147e6fbc3c	VirusShare_8725e4888cd76146280a801c8bfa8f4a	5a87af64d1ed0c8cc...	Sat May 11 21:08:45 2024	Sat May 11 21:10:11 2024	86.2496161460876	1	Pass	211883	33364	NtCancelTimer	NtAllocateVirtualMemory

5.11 pav. Galutinio etapo „Session“ duomenų bazės pavyzdys

Sisteminių kreipinių kiekis sulėmė 8,5 GB duomenų bazės failo apimtį. Taip yra dėl to, nes iš visų gautų sisteminių kreipinių, atliekant filtravimą, buvo išgauti tik aktualūs sisteminiai kreipiniai. Atlikus filtravimą, sisteminių kreipinių yra 6,78 karto mažiau. Atitinkamai, jeigu norime sužinoti apytikslį duomenų bazės dydi, jei sisteminiai kreipiniai nebūtų filtruojami, turime 8,5 GB padauginti iš 6,78298. Gaunamas rezultatas yra ~ 57,65 GB. Tačiau reikia pabrėžti, kad kalbama apie „Drak Management“ duomenų bazės dydį. „Drakvuf Sandbox“ sisteminių kreipinių analizės KPK pavyzdžio žurnalizuotas failas turi daugiau parametrų, kurie nebuvo įtraukiami į „Drak Management“

įrankio kuriamą duomenų bazę. Esant poreikiui, visuomet galima atlikti pakeitimus „Drak Management“ kodo struktūroje, funkcijose ir įtraukti trūkstamus sisteminių kreipinių parametrus. Atlikto finalinio eksperimento su 1000 KPK pavyzdžių sisteminių kreipinių kiekių rezultatai pateikiami 5.6 lentelėje.

5.6 lentelė. Galutinio etapo tyrimo metu surinktų sisteminių kreipinių statistinė lentelė

Nefiltruotų sisteminių kreipinių kiekis	146 774 183
Filtruotų sisteminių kreipinių kiekis	21 638 592

Lentelėje pateikti duomenys yra suminiai, tačiau „Drak Management“ pateikia plačią kiekvieno KPK pavyzdžio išklotinę ir tai leidžia nustatyti sisteminių kreipinių dažnį. Grafiko iliustracijoje (pav. 5.12) pateikiami tokie duomenys:

- Horizontali ašis – išrikiuoti KPK pavyzdžiai nuo 1 iki 1000.
- Vertikali ašis – sisteminių kreipinių, kuriuos pavyko užfiksuoti, kiekis.
- Geltona linija ir taškai pažymi nefiltruotus sisteminių kreipinių kiekius.
- Raudona linija ir taškai pažymi filtruotus sisteminių kreipinių kiekius.



5.12 pav. Galutinio etapo nefiltruotų ir filtruotų sisteminių kreipinių dažniai

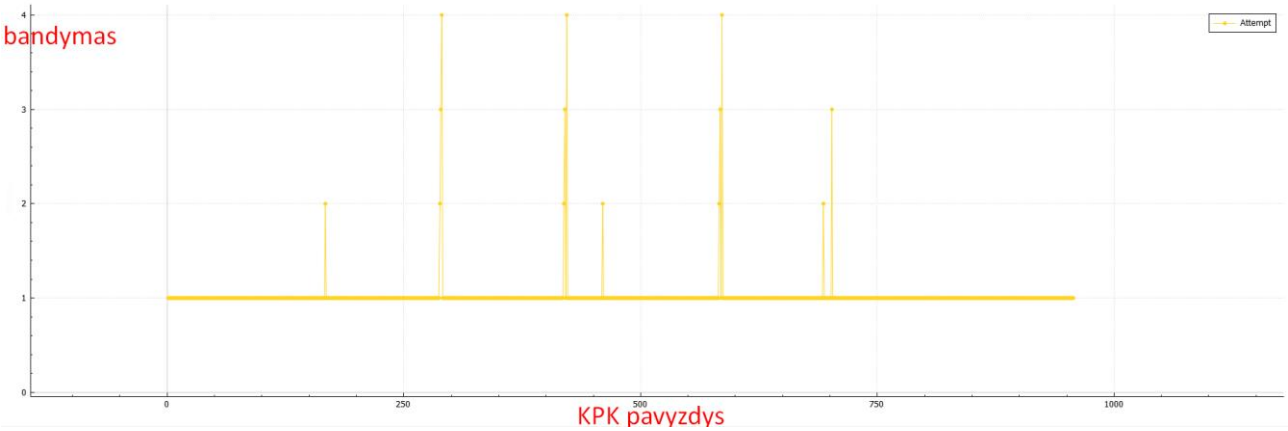
Lentelėje 5.7 pateikiami surinkti tiek filtruotų, tiek nefiltruotų sisteminių kreipinių vieno KPK pavyzdžio analizės metu statistiniai duomenys.

5.7 lentelė. Galutinio etapo tyrimo metu surinktų vieno KPK pavyzdžio analizės sisteminių kreipinių kiekio statistinė lentelė

Daugiausiai surinktų nefiltruotų sisteminių kreipinių	604 737
Mažiausiai surinktų nefiltruotų sisteminių kreipinių	16 991
Nefiltruotų sisteminių kreipinių vidurkis	168 706

Daugiausiai surinktų filtruotų sisteminių kreipinių	425 517
Mažiausiai surinktų filtruotų sisteminių kreipinių	2
Filtruotų sisteminių kreipinių vidurkis	24872

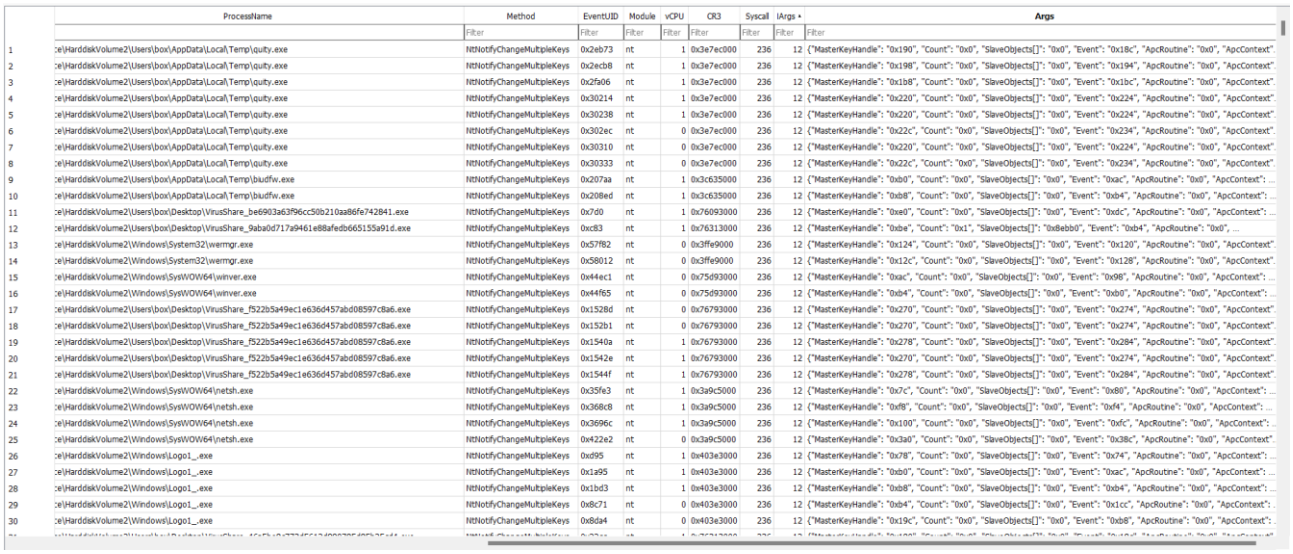
Taip pat 1000 KPK pavyzdžių finalinio empirinio tyrimo etape pasitaikė atvejų, kuomet KPK tyrimas buvo kartojamas iki 4 kartų (pav. 5.13). Šie pavieniai atvejai rodo, kad „Drakvuf“ įrankiui nepavyko užmegzti tinkamo ryšio su „Guest“ VM iš pirmo bandymo ir perduoti KPK pavyzdį testavimui.



5.13 pav. „Drak Management“ bandymų išanalizuoti KPK pavyzdį kiekis

Kaip buvo aiškinama ankstesniame skyriuje, „Drakvuf Sandbox“ perduoda KPK „Drakvuf“ įrankiui, o „Drakvuf“ KPK įkėlimą atlieka iki trijų kartų. Šio tyrimo naudojamoje konfigūracijoje „Drak Management“, priklausomai, kaip tyrėjas sukonfigūravo įrankį, galėjo atlikti KPK tyrimą (kiekvienas naujas tyrimas įgauna nauja identifikacinį numerį) iki 4 kartų. Būtent todėl 6.10 iliustracijoje didžiausias bandymų kiekis buvo – 4.

Didžiausias perduodamas kiekis argumentų – 12. Tai buvo vykdoma per „NtNotifyChangeMultipleKeys“ sisteminį kreipinį (pav. 5.14). Taip pat žemiau pateikiamame pavyzdyje galima įžvelgti ir dar anksčiau nematytų procesų pavadinimų. Kai atliekamas KPK perkėlimas į „Guest“ VM, kenkėjiškai programai yra priskiriamas proceso identifikacinis numeris. Paleistas KPK gali kurti dukterinius procesus – toks reiškinys nėra naujas ir yra būdingas kenkėjiškoms programoms, ypač tada, kuomet jos bando paslėpti kenkėjiškus ketinimus kurdamos naujus (dukterinius) procesus. Dukteriniai procesai visuomet turi tiek savo, tiek tėvinį identifikacinį numerį, kuris padeda atsekti pradinį procesą, kuris kuria dukterinius procesus. Remiantis šia logika, „Drak Management“ algoritmas atlieka filtravimą.



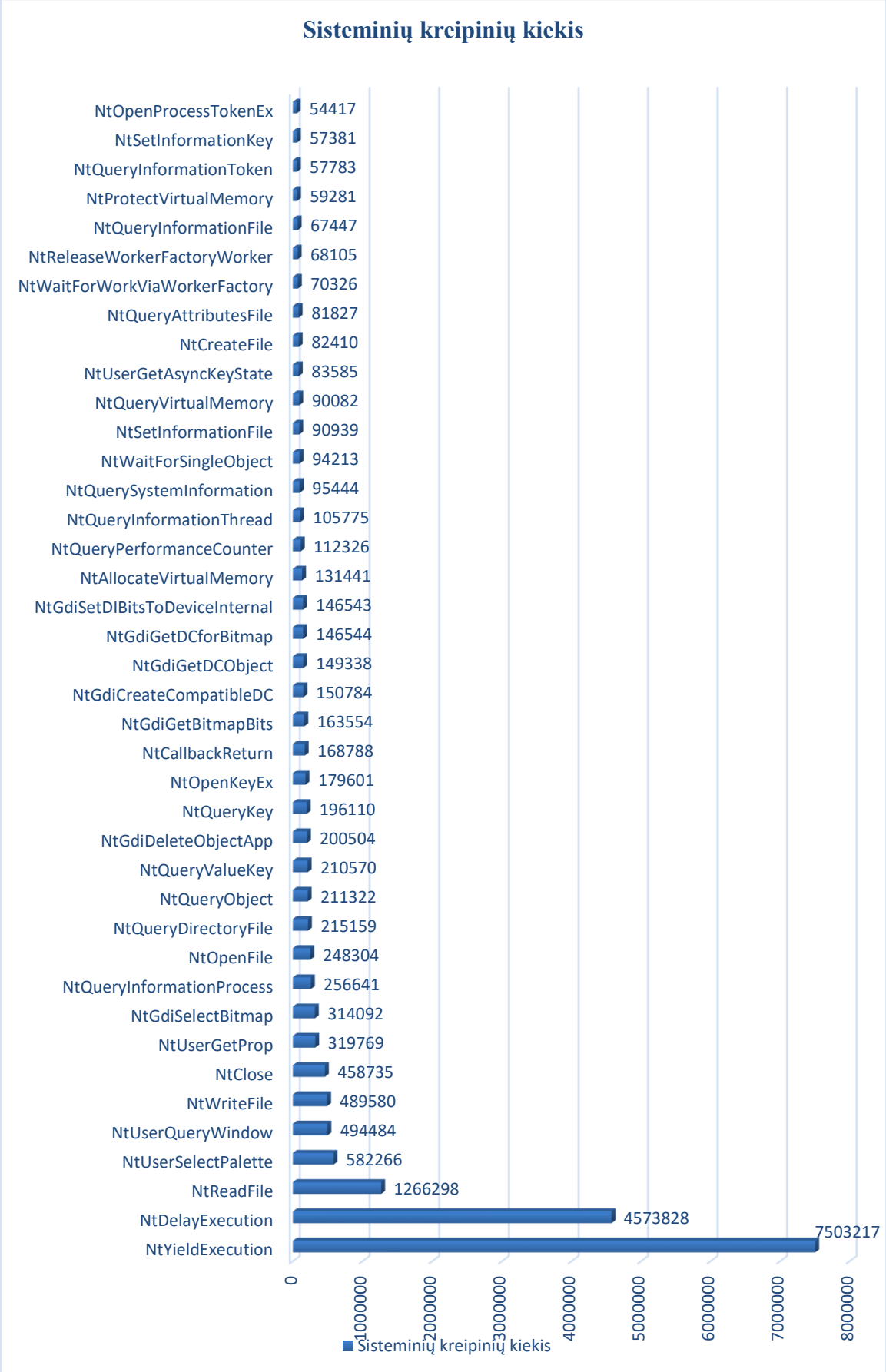
	ProcessName	Method	EventUID	Module	vCPU	CR3	Syscall	lArgs	Args
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	:e\HarddiskVolume2\Users\box\AppData\Local\Temp\quity.exe	NtNotifyChangeMultipleKeys	0x2eb73	nt	1	0x3e7ec000	236	12	{"MasterKeyHandle": "0x190", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0x18c", "ApcRoutine": "0x0", "ApcContext"
2	:e\HarddiskVolume2\Users\box\AppData\Local\Temp\quity.exe	NtNotifyChangeMultipleKeys	0x2ecb8	nt	1	0x3e7ec000	236	12	{"MasterKeyHandle": "0x198", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0x194", "ApcRoutine": "0x0", "ApcContext"
3	:e\HarddiskVolume2\Users\box\AppData\Local\Temp\quity.exe	NtNotifyChangeMultipleKeys	0x2fa06	nt	1	0x3e7ec000	236	12	{"MasterKeyHandle": "0x1b8", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0x1bc", "ApcRoutine": "0x0", "ApcContext"
4	:e\HarddiskVolume2\Users\box\AppData\Local\Temp\quity.exe	NtNotifyChangeMultipleKeys	0x30214	nt	1	0x3e7ec000	236	12	{"MasterKeyHandle": "0x220", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0x224", "ApcRoutine": "0x0", "ApcContext"
5	:e\HarddiskVolume2\Users\box\AppData\Local\Temp\quity.exe	NtNotifyChangeMultipleKeys	0x30238	nt	1	0x3e7ec000	236	12	{"MasterKeyHandle": "0x220", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0x224", "ApcRoutine": "0x0", "ApcContext"
6	:e\HarddiskVolume2\Users\box\AppData\Local\Temp\quity.exe	NtNotifyChangeMultipleKeys	0x302ec	nt	0	0x3e7ec000	236	12	{"MasterKeyHandle": "0x22c", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0x234", "ApcRoutine": "0x0", "ApcContext"
7	:e\HarddiskVolume2\Users\box\AppData\Local\Temp\quity.exe	NtNotifyChangeMultipleKeys	0x30310	nt	0	0x3e7ec000	236	12	{"MasterKeyHandle": "0x220", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0x224", "ApcRoutine": "0x0", "ApcContext"
8	:e\HarddiskVolume2\Users\box\AppData\Local\Temp\quity.exe	NtNotifyChangeMultipleKeys	0x30333	nt	0	0x3e7ec000	236	12	{"MasterKeyHandle": "0x22c", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0x234", "ApcRoutine": "0x0", "ApcContext"
9	:e\HarddiskVolume2\Users\box\AppData\Local\Temp\biudfw.exe	NtNotifyChangeMultipleKeys	0x207aa	nt	1	0x3c635000	236	12	{"MasterKeyHandle": "0xb0", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0xac", "ApcRoutine": "0x0", "ApcContext": ...
10	:e\HarddiskVolume2\Users\box\AppData\Local\Temp\biudfw.exe	NtNotifyChangeMultipleKeys	0x208ed	nt	1	0x3c635000	236	12	{"MasterKeyHandle": "0xb8", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0xb4", "ApcRoutine": "0x0", "ApcContext": ...
11	:e\HarddiskVolume2\Users\box\Desktop\VirusShare_be6903a63f96cc50b210aa86fe742841.exe	NtNotifyChangeMultipleKeys	0x7d0	nt	1	0x76093000	236	12	{"MasterKeyHandle": "0xe0", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0xdc", "ApcRoutine": "0x0", "ApcContext": ...
12	:e\HarddiskVolume2\Users\box\Desktop\VirusShare_9aba0d717a9461e88afedb665155a91d.exe	NtNotifyChangeMultipleKeys	0xc83	nt	1	0x76313000	236	12	{"MasterKeyHandle": "0xbe", "Count": "0x1", "SlaveObjects[]": "0x8ebb0", "Event": "0xb4", "ApcRoutine": "0x0", ...
13	:e\HarddiskVolume2\Windows\System32\wermgr.exe	NtNotifyChangeMultipleKeys	0x57f82	nt	0	0x3ffe9000	236	12	{"MasterKeyHandle": "0x124", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0x120", "ApcRoutine": "0x0", "ApcContext"
14	:e\HarddiskVolume2\Windows\System32\wermgr.exe	NtNotifyChangeMultipleKeys	0x58012	nt	0	0x3ffe9000	236	12	{"MasterKeyHandle": "0x12c", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0x128", "ApcRoutine": "0x0", "ApcContext"
15	:e\HarddiskVolume2\Windows\SysWOW64\winver.exe	NtNotifyChangeMultipleKeys	0x44ec1	nt	0	0x75d93000	236	12	{"MasterKeyHandle": "0xac", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0x98", "ApcRoutine": "0x0", "ApcContext": ...
16	:e\HarddiskVolume2\Windows\SysWOW64\winver.exe	NtNotifyChangeMultipleKeys	0x44f65	nt	0	0x75d93000	236	12	{"MasterKeyHandle": "0xb4", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0xb0", "ApcRoutine": "0x0", "ApcContext": ...
17	:e\HarddiskVolume2\Users\box\Desktop\VirusShare_f522b5a49ec1e636d457abd08597c8a6.exe	NtNotifyChangeMultipleKeys	0x1528d	nt	0	0x76793000	236	12	{"MasterKeyHandle": "0x270", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0x274", "ApcRoutine": "0x0", "ApcContext"
18	:e\HarddiskVolume2\Users\box\Desktop\VirusShare_f522b5a49ec1e636d457abd08597c8a6.exe	NtNotifyChangeMultipleKeys	0x152b1	nt	0	0x76793000	236	12	{"MasterKeyHandle": "0x270", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0x274", "ApcRoutine": "0x0", "ApcContext"
19	:e\HarddiskVolume2\Users\box\Desktop\VirusShare_f522b5a49ec1e636d457abd08597c8a6.exe	NtNotifyChangeMultipleKeys	0x1540a	nt	1	0x76793000	236	12	{"MasterKeyHandle": "0x278", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0x284", "ApcRoutine": "0x0", "ApcContext"
20	:e\HarddiskVolume2\Users\box\Desktop\VirusShare_f522b5a49ec1e636d457abd08597c8a6.exe	NtNotifyChangeMultipleKeys	0x1542e	nt	1	0x76793000	236	12	{"MasterKeyHandle": "0x270", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0x274", "ApcRoutine": "0x0", "ApcContext"
21	:e\HarddiskVolume2\Users\box\Desktop\VirusShare_f522b5a49ec1e636d457abd08597c8a6.exe	NtNotifyChangeMultipleKeys	0x1544f	nt	1	0x76793000	236	12	{"MasterKeyHandle": "0x278", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0x284", "ApcRoutine": "0x0", "ApcContext"
22	:e\HarddiskVolume2\Windows\SysWOW64\netsh.exe	NtNotifyChangeMultipleKeys	0x35fe3	nt	1	0x3a9c5000	236	12	{"MasterKeyHandle": "0x7c", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0x80", "ApcRoutine": "0x0", "ApcContext": ...
23	:e\HarddiskVolume2\Windows\SysWOW64\netsh.exe	NtNotifyChangeMultipleKeys	0x368c8	nt	1	0x3a9c5000	236	12	{"MasterKeyHandle": "0xf8", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0xf4", "ApcRoutine": "0x0", "ApcContext": ...
24	:e\HarddiskVolume2\Windows\SysWOW64\netsh.exe	NtNotifyChangeMultipleKeys	0x3696c	nt	1	0x3a9c5000	236	12	{"MasterKeyHandle": "0x100", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0xfc", "ApcRoutine": "0x0", "ApcContext": ...
25	:e\HarddiskVolume2\Windows\SysWOW64\netsh.exe	NtNotifyChangeMultipleKeys	0x422e2	nt	0	0x3a9c5000	236	12	{"MasterKeyHandle": "0x3a0", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0x38c", "ApcRoutine": "0x0", "ApcContext"
26	:e\HarddiskVolume2\Windows\Logo1_.exe	NtNotifyChangeMultipleKeys	0xd95	nt	1	0x403e3000	236	12	{"MasterKeyHandle": "0x78", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0x74", "ApcRoutine": "0x0", "ApcContext": ...
27	:e\HarddiskVolume2\Windows\Logo1_.exe	NtNotifyChangeMultipleKeys	0x1a95	nt	1	0x403e3000	236	12	{"MasterKeyHandle": "0xb0", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0xac", "ApcRoutine": "0x0", "ApcContext": ...
28	:e\HarddiskVolume2\Windows\Logo1_.exe	NtNotifyChangeMultipleKeys	0x1bd3	nt	1	0x403e3000	236	12	{"MasterKeyHandle": "0xb8", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0xb4", "ApcRoutine": "0x0", "ApcContext": ...
29	:e\HarddiskVolume2\Windows\Logo1_.exe	NtNotifyChangeMultipleKeys	0x8c71	nt	0	0x403e3000	236	12	{"MasterKeyHandle": "0xb4", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0x1cc", "ApcRoutine": "0x0", "ApcContext": ...
30	:e\HarddiskVolume2\Windows\Logo1_.exe	NtNotifyChangeMultipleKeys	0x8da4	nt	0	0x403e3000	236	12	{"MasterKeyHandle": "0x19c", "Count": "0x0", "SlaveObjects[]": "0x0", "Event": "0xb8", "ApcRoutine": "0x0", "ApcContext": ...

5.14 pav. Didžiausias kiekis sisteminio kreipinio perduodamų argumentų

Toliau pateikiami analizės metu išgauti vieni unikalesnių dukterinių procesų pavadinimai, kurie buvo iššaukiami tėvinio proceso. Visas 328 procesų (neskaitant tėvinių procesų su pavadinimais „VirusShare...“) sąrašas pridedamas šio rašto darbo **Prieduose (8)**.

_v_5539.exe	*edkugeo.exe*	*net.exe*	*sc.exe*
alg.exe	*exc.exe*	*netsh.exe*	*schtasks.exe*
bcdedit.exe	*explorer.exe*	*niwtgsgiq.exe*	*srtsrv32.exe*
biudfw.exe	*facebook updat*	*nkfwaqbxrgaoy.*	*tacre.exe*
budha.exe	*hcbnaf.exe*	*oMcwMcQI.exe*	*virusshare_93a*
bzfl.exe	*huter.exe*	*outlook.exe*	*vssadmin.exe*
cmd.exe	*jusched.exe*	*peuineb.exe*	*wermgr.exe*
codecupdate.ex	*kygEYUwY.exe*	*q.exe*	*wevtutil.exe*
conhost.exe	*lasma.exe*	*quip.exe*	*winlogon.exe*
copvvtyjpfenkn	*lou.exe*	*quity.exe*	*winver.exe*
ctfmon.exe	*lssmon.exe*	*reg.exe*	*wpujcvnd.exe*
dw20.exe	*nbjdxqvkjh.exe*	*sander.exe*	*wytyg.exe*

Iš viso buvo surinkti 395 unikalūs sisteminiai kreipiniai, kurie visų bendras iškviečiamų kiekis yra 21638592 kartus. 40 dažniausiai iškviečiamų sisteminių kreipinių sąrašas pateikiamas grafike (pav. 5.15). Joje horizontaliąją ašį atspindi sisteminių kreipinių kiekis, o vertikaliąją ašyje išvardinti sisteminiai kreipiniai.



5.15 pav. Galutinio etapo 40 daugiausiai surinktų sisteminių kreipinių dažnumas

Labiausiai išskirtinas sisteminis kreipinys – „NtYieldExecution“ su 7,5 mln. iškvietimų. Lygiai tas pats sisteminis kreipinys dominavo ir tarpiniame greitaveikos tyrimo etape. Toliau seka „NtDelayExecution“, „NtReadFile“, „NtUserSelectPalette“ bei „NtUserQueryWindow“.

Galutinio tyrimo etapo metu buvo išgauti sisteminiai kreipiniai, iš kurių galima išskirti didžiausią sąsają su KPK turinčių sisteminius kreipinius remiantis mokslinių straipsnių analize. Tai pateikiama 5.8 lentelėje.

5.8 lentelė. Galutinio tyrimo etapo metu labiausiai su KPK asocijuojamų išgautų kreipinių kiekiai

Sisteminis kreipinys	Sąsaja su KPK	Kiekis
„NtCreateFile“	Dažniausiai naudojami atidaryti esamiems nei naujiems failams kurti, platinti kenkėjišką kodą.	82410
„NtOpenFile“		248304
„NtReadFile“	Nustaiko / rašo į failus, KPK gali modifikuoti failo turinį, tokiu būdu užkrėsti kitus failus.	1266298
„NtWriteFile“		489580
„NtMapViewOfSection“	Dažniausiai naudojama atliekant KPK injekcijas į nepažeistus procesus, vadinama „process hollowing“ (MDSec Research, 2020).	49918
„NtUnmapViewOfSection“		28127
„NtCreateSection“	Kuria, atidaro kritiškai svarbius atminties valdymui sekcijos objektus – „process hollowing“ technika (Gavriel, 2018).	18384
„NtOpenSection“		20056
„NtAllocateVirtualMemory“	Naudojama virtualiai atminčiai paskirstyti, atlaisvinti siekiant įterpti KPK į atmintį bei išvengti aptikimo	131441
„NtFreeVirtualMemory		47902
„NtQueryInformationProcess“	Nuskaito proceso informaciją, įskaitant, ar yra naudojamas „debugging“ įrankis, kad galėtų nuslėpti kenkėjiškąsias savybes (Ayman, 2021).	256641
„NtCreateUserProcess“	Naujų procesų bei gijų kūrimui. Suteikia galimybę kurti kenkėjiškus procesus arba atlikti „process hollowinng“.	622
„NtCreateThreadEx“		2176
„NtClose“	Uždaro sąsajas su įvairiais procesais, gijomis, failais. Tai leidžia kenkėjiškai programai palikti mažiau pėdsakų.	458735

Taigi šiame skyriuje buvo apžvelgti visų empirinio tyrimo etapų rezultatai, todėl galima apibendrinti šį skyrių su palyginamąja lentele (5.9), kurioje palyginamas „Drak Management“ įrankio su VMI technika įgyvendintas dinaminės KPK elgsenos rinkimo patobulinimas ir D. Čeponio AWSCTD rinkinio sudarymo metodas.

5.9 lentelė. Šio empirinio tyrimo rezultatų ir AWSCTD kūrimo metodo rezultatų palyginimas

Kriterijus	AWSCTD metodas	„Drak Management“ su VMI technika patobulinimas
KPK rinkinio sudarymo data	2017 m. 06 mėn.	2024 m. 04 mėn.
KPK imtis	12 100	1000
Trukmė	~ 2 mėn. (~ 1488 val.)	37 val. 15 min. 22 sek.
„Guest“ VM kiekis	6	2
„Guest“ OS	„Windows 7“	„Windows 7“
Vidutinė trukmė 1 KPK analizei	442 sek.	125 sek. (3,5 kartų greičiau) galutinio tyrimo
Vidutinė trukmė 1 KPK analizei	442 sek.	85 sek. (5,2 kartų greičiau) greitaveikos tyrimo
Sisteminių (sis.) kreipinių (kr.) kiekis	112,56 mln.	21,63 mln.
Sėkmės rodiklis	89,34 %	76,84 %
1000 KPK imties sis. kr. kiekis	= 9,30 mln.	21,63 mln. (2,32 kartų daugiau)

Išvados

1. Dinaminis KPK aptikimas gali būti atliekamas trimis pagrindiniais metodais: statine, dinamine ar hibridine analize. Kiekvienas iš šių metodų turi savų privalumų ir trūkumų, kurie buvo išsamiai analizuoti. Tyrimai parodė, kad pažangios kenkėjiškos programos gali išvengti aptikimo tradiciniais metodais paremtose virtualizacijos aplinkose, todėl yra būtina tobulinti dinaminės KPK elgsenos rinkimo metodus, siekiant efektyviau analizuoti KPK.
2. VMI (angl. *Virtual Machine Introspection*) technika leidžia tiesiogiai analizuoti ir modifikuoti virtualių mašinų (VM) atmintį ir būseną. Tai suteikia galimybę gauti vertingą informaciją apie kenkėjiškų programų elgesį ir pagerinti KPK analizę. Tyrimai parodė, kad naudojant tinkamus įrankius, tokius kaip „LibVMI“ ir „Drakvuf“, galima pasiekti didesnį našumą ir duomenų saugumą analizuojant kenkėjiškas programas.
3. Palyginus „Cuckoo“ ir „Drakvuf“ įrankius, nustatyta, kad VMI pagrįsti įrankiai, tokie kaip „Drakvuf“, užtikrina greitesnę KPK analizę, mažesnę saugyklos apkrovą ir daugiau duomenų apie KPK veiklą. VMI technika ypač naudinga stebint visus vartotojo ir branduolio lygmenyje iškviečiamus sisteminius kreipinius.
4. Atlikus „Drakvuf“ mokslinės literatūros analizę ir D. Čeponio „DrStrace“ su „ProxMox“ įgyvendinto sisteminių kreipinių rinkimo metodo analizę, galima teigti, jog panaudojus VMI technika pasižyminčius įrankius, įžvelgiamas potencialas praplėsti AWSCTD duomenų rinkinį pritaikant kitokią „Windows“ sisteminių kreipinių rinkimo techniką.
5. Pasiūlyti patobulinimai, taikant VMI techniką, padidino KPK elgsenos rinkimo efektyvumą. Naujai sukurtas „Drak Management“ įrankis automatizuoja procesus, palengvina tyrėjo darbą ir efektyviai renka bei apdoroja duomenis. Šie patobulinimai leido surinkti didesnį kiekį tikslesnių sisteminių kreipinių per trumpesnį analizės laiką.
6. Empirinio tyrimo rezultatai patvirtino VMI pranašumus, rodydami, kad KPK analizė tampa 4-5 kartus efektyvesnė, išgaunama daugiau sisteminių kreipinių. Tačiau „Drakvuf Sandbox“ dar nepalaiko naujausių „Windows“ OS, todėl vykdant tyrimus su „Windows 7“ OS, sėkmės rodiklis buvo ~ 80 % dėl KPK ir OS nesuderinamumo naudojant tuo metu naujausią KPK pavyzdžių rinkinį.

Literatūros sąrašas

- Ayman D. (2021). *Anti-Debugging Techniques* https://medium.com/@X3non_C0der/anti-debugging-techniques-eda1868e0503
- Alenezi, M. N., Alabdulrazzaq, H., Alshaher, A. A., & Alkharang, M. M. (2020). *Evolution of malware threats and techniques: A review. International journal of communication networks and information security, 12(3), 326-337*. https://www.researchgate.net/profile/Haneen-Alabdulrazzaq/publication/349324759_Evolution_of_Malware_Threats_and_Techniques_A_Review/links/602ad0e64585158939a93934/Evolution-of-Malware-Threats-and-Techniques-A-Review.pdf
- Aleroud, A., & Zhou, L. (2017). Phishing environments, techniques, and countermeasures: A survey. Computers & Security, 68, 160-196. https://www.sciencedirect.com/science/article/abs/pii/S0167404817300810
- Baker K. (2022). *Malware Analysis*. https://www.crowdstrike.com/cybersecurity-101/malware/malware-analysis/
- Bedrocksystems.com (2020). *An Introduction to Virtual Machine Introspection* https://bedrocksystems.com/virtual-machine-introspection/
- Belding G. (2020). *Virtualization-based sandbox malware*. https://resources.infosecinstitute.com/topic/virtualization-based-sandbox-malware/
- Bitdefender (2023). *libkvmi* https://github.com/bitdefender/libkvmi
- Buxton O. (2021). *How to Delete Temporary Files in Windows 10, 8, & 7* https://www.avast.com/c-delete-temporary-files-windows
- CERT Polska (2021). *DRAKVUF Sandbox Documentation* https://drakvuf-sandbox.readthedocs.io/en/latest/index.html#
- Cisco. (n.d.). *What Is Malware?* https://www.cisco.com/c/en_au/products/security/advanced-malware-protection/what-is-malware.html
- Čeponis D. (2020). *AWSCTD* https://github.com/DjPasco/AWSCTD
- Čeponis D., Goranin N. (2018). *Towards a Robust Method of Dataset Generation of Malicious Activity for Anomaly-Based HIDS Training and Presentation of AWSCTD Dataset*. https://www.bjmc.lu.lv/fileadmin/user_upload/lu_portal/projekti/bjmc/Contents/6_3_01_Ceponis.pdf

- Dangl T., Taubmann B., Reiser H. P. (2021). *RapidVMI: Fast and multi-core aware active virtual machine introspection* https://dl.acm.org/doi/10.1145/3465481.3465752
- Egele, M., Kruegel, C., Kirda, E., Yin, H., & Song, D. (2007). *Dynamic spyware analysis*. https://www.usenix.org/legacy/events/usenix07/tech/full_papers/egele/egele.pdf
- Fox N. (2021). *11 Best Malware Analysis Tools and Their Features | Varonis*. https://www.varonis.com/blog/malware-analysis-tools
- Gandotra E., Bansal D., Sofat S. (2014). *Malware Analysis and Classification: A Survey*. https://www.scirp.org/html/4-7800194_44440.htm
- Garfinkel T., Rosenblum M. (2003). *A Virtual Machine Introspection Based Architecture for Intrusion Detection* http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.11.8367&rep=rep1&type=pdf
- Gavriel H. (2018). *Malware Mitigation when Direct System Calls are Used* https://www.cyberbit.com/endpoint-security/malware-mitigation-when-direct-system-calls-are-used/
- Google (2020). rekall https://github.com/google/rekall
- Grappendorf D., Bremicker S., Roy A., Lausser G., Hutty D., Schäfer D., Brogi N. (2016). *Vagrant Proxmox Provider* https://github.com/telcat/vagrant-proxmox
- Gupta S., Sharma H., Kaur S. (2018). *Malware Characterization Using Windows API Call Sequences.* https://journals.riverpublishers.com/index.php/JCSANDM/article/download/5309/3877/
- Hynry1679 (2020). *Windows 7 Professional SP1 (32 bit and 64 bit ISOs)* https://archive.org/details/win-7-pro-32-64-iso
- ytisf (2017). *theZoo* https://github.com/ytisf/theZoo
- Lastline (2018). *Virtualization-Based Sandboxes are Vulnerable to Advanced Malware*. https://www.lastline.com/blog/virtualization-based-sandboxes/
- Lengyel, T. K., Maresca, S., Payne B. D., Webster G. D., Vogl S., Kiayias A. (2014). *Scalability, Fidelity and Stealth in the DRAKVUF Dynamic Malware Analysis System* https://doi.org/10.1145/2664243.2664252
- Li B., Li J. (2011). *A VMM-Based System Call Interposition Framework for Program Monitoring* https://www.researchgate.net/publication/224213377_A_VMM-Based_System_Call_Interposition_Framework_for_Program_Monitoring
- Libvmi.com (n.d.). *Introduction to LibVMI* https://libvmi.com/docs/gcode-intro.html

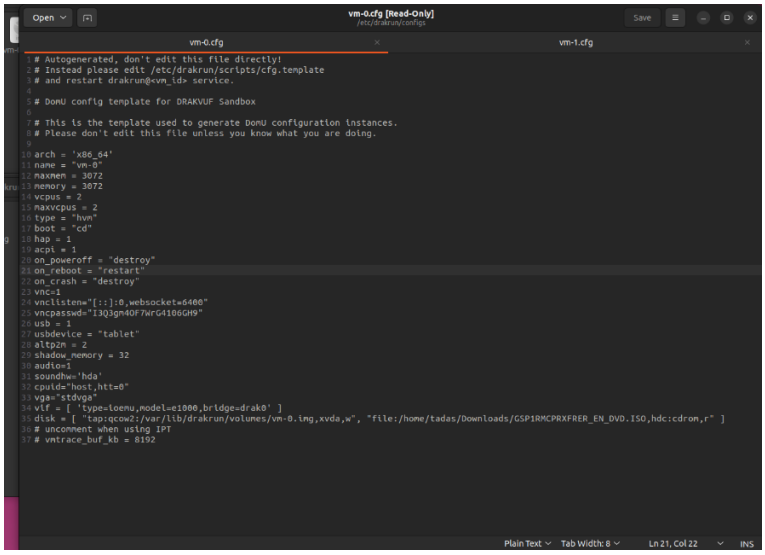
- MDSec Research (2020). *Bypassing User-Mode Hooks and Direct Invocation of System Calls for Red Teams* https://www.mdsec.co.uk/2020/12/bypassing-user-mode-hooks-and-direct-invocation-of-system-calls-for-red-teams/
- Melvin A., Kathrine G. J. (2021). *A Quest for Best: A Detailed Comparison Between Drakvuf-VMI-Based and Cuckoo Sandbox-Based Technique for Dynamic Malware Analysis* https://www.researchgate.net/publication/343223298_A_Quest_for_Best_A_Detailed_Comparison_Between_Drakvuf-VMI-Based_and_Cuckoo_Sandbox-Based_Technique_for_Dynamic_Malware_Analysis
- More A., Tapaswi S. (2014). *Virtual machine introspection: towards bridging the semantic gap* https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-014-0016-2
- Nacionalinis Kibernetinio Saugumo Centras. (n.d.). *Phishing*. https://www.nksc.lt/rekomendacijos/phishing.html
- Purplesec.us (2021). *2021 Cyber Security Statistics. The Ultimate List Of Stats, Data & Trends.* https://purplesec.us/resources/cyber-security-statistics/
- Samsclass.info. (n.d.). *Proj 14: Malware Behavior (Lab 11-1) (35 pts.)* https://samsclass.info/126/proj/p14-lab11-1
- Sihwail R., Omar K., Ariffin K. A. Z. (2018). *A Survey on Malware Analysis Techniques: Static, Dynamic, Hybrid and Memory Analysis*. https://core.ac.uk/download/pdf/325990564.pdf
- SQLite (2022). https://www.sqlite.org/draft/index.html
- StatCounter (n.d.). *Desktop Windows Version Market Share Worldwide* https://gs.statcounter.com/windows-version-market-share/desktop/worldwide/#monthly-201707-202404
- Stichting Cuckoo Foundation (2023). *Cuckoo*. https://cuckoosandbox.org/
- Stichting Cuckoo Foundation. (2019). *Cuckoo FAQ*. https://cuckoo.sh/docs/introduction/what.html
- Strimaitis V. (n.d.). *Simbolių eilutės.* https://vstrimaitis.github.io/siom-cpp/simboliu-eilutes/
- Švabauskas T. (2024). *„Drak-Management“ įrankio demonstracija* https://www.youtube.com/watch?v=PZrxU8YJlkg
- Švabauskas T. (2024). *„Drak-Management“ įrankis* https://github.com/tadassv/drak-management

- Švabauskas T. (2024). *1000 KPK tyrimo rezultatų failai* https://drive.google.com/drive/folders/1w5_roF5qk-1XYO-_mT97kQ5jz7GyrYD7
- Švabauskas T. (2024). *Trumpa „Drak-Management“ įrankio demonstracija* https://youtu.be/nE5lNIz19Vw
- Ubuntu documentation (n.d.). *Xen* https://help.ubuntu.com/community/Xen
- VirusShare BitTorrent Tracker (n.d.). *VirusShare_00289.zip* http://tracker.virusshare.com:6969/details.tmpl?torrent_info_hash=9374576563AB55F8CAB1F0A63372673FD4D0E4BD
- VirusShare BitTorrent Tracker (n.d.). *VirusShare_00486.zip* http://71.105.224.114:6969/details.tmpl?torrent_info_hash=DAC78EE9B8AD6EA0B773FA5693167BE69AE02AAA
- Wang L., Wang B., Liu J., Miao Q., Zhang J. (2019). *Cuckoo-based Malware Dynamic Analysis [J]. Int J Performability Eng, 2019, 15(3): 772-781* http://www.ijpe-online.com/EN/10.23940/ijpe.19.03.p6.772781
- Xiaofeng L., Xiao Z., Fangshuo J., Shengwei Y., Jing S. (2018). *ASSCA: API based Sequence and Statistics features Combined malware detection Architecture, ISSN 1877-0509.* https://www.sciencedirect.com/science/article/pii/S1877050918302965
- Zhang Q., Reeves D. S. (2007). *MetaAware: Identifying Metamorphic Malware. Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007).* https://ieeexplore.ieee.org/abstract/document/4413007

Priedai

Priedas nr. 1 – „Guest“ VM konfigūracijos failas

Šiuo atveju pateikiama pradinės VM būsenos konfigūracija, kuri nurodo įvairius VM parametrus, pavyzdžiui, jos architektūrą („x86_64“), pavadinimą ("vm-0"), atminties kiekį ir naudojamą procesoriaus branduolius bei įvairius kitus parametrus, susijusius su paleidimu, valdymu ir tinklų kūrimu. „VNC“ prieigos slaptažodis yra pateikiamas šioje konfigūracijoje.



Priedas nr. 2 – „Guest“ VM pradinio atvaizdo modifikacijos

1. Sustabdomi „drakrun“ servisai, paleidžiama „drakplayground 0“

```
root@tadas-MS-7984: /home/tadas
root@tadas-MS-7984: /home/tadas
root@tadas-MS-7984:/home/tadas# xl list
Name                                        ID   Mem VCPUs      State   Time(s)
Domain-0                                     0  8169     2     r-----      88.0
root@tadas-MS-7984:/home/tadas# systemctl stop drakrun@*
root@tadas-MS-7984:/home/tadas# drak-healthcheck
Checking daemon status...
drak-web.service            OK
drak-system.service         OK
drak-minio.service          OK
Checking worker status...
drakrun@2.service           ERROR
drakrun@1.service           ERROR
drak-postprocess@1.service  OK
root@tadas-MS-7984:/home/tadas# drak-healthcheck
Checking daemon status...
drak-web.service            OK
drak-system.service         OK
drak-minio.service          OK
Checking worker status...
drakrun@2.service           ERROR
drakrun@1.service           ERROR
drak-postprocess@1.service  OK
root@tadas-MS-7984:/home/tadas# drakplayground 0
[2024-05-06 20:37:00,456][INFO] Ensuring that drakrun@0 service is stopped...
[2024-05-06 20:37:00,495][INFO] Generated VM configuration for vm-0
[2024-05-06 20:37:00,525][DEBUG] iptables chains already exist, no setup needed
[2024-05-06 20:37:00,537][INFO] Created bridge drak0
[2024-05-06 20:37:00,577][INFO] Bridge drak0 is up
[2024-05-06 20:37:00,700][INFO] Restoring VM vm-0
Loading new save file /var/lib/drakrun/volumes/snapshot.sav (new xl fmt info 0x3
/0x0/2033)
 Savefile contains xl domain config in JSON format
Parsing config from /etc/drakrun/configs/vm-0.cfg
xc: info: Found x86 HVM domain from Xen 4.16
xc: info: Restoring domain
xc: info: Restore successful
xc: info: XenStore: mfn 0xfeffc, dom 0, evt 1
xc: info: Console: mfn 0xfefff, dom 0, evt 2
Dnsmasq version 2.90  Copyright (c) 2000-2024 Simon Kelley
Compile time options: IPv6 GNU-getopt DBus no-UBus i18n IDN DHCP DHCPv6 no-Lua T
FTP conntrack ipset no-nftset auth cryptohash DNSSEC loop-detect inotify dumpfil
e

This software comes with ABSOLUTELY NO WARRANTY.
Dnsmasq is free software, and you are welcome to redistribute it
under the terms of the GNU General Public License, version 2 or 3.
[2024-05-06 20:37:02,933][DEBUG] Using selector: EpollSelector
```

2. Įkeliama testavimui skirta „Windows“ vykdomoji programa, tuomet atliekami veiksmai prisijungus su „VNCviewer“ į „Guest“ aplinką. Pabaigus konfigūruoti, vykdoma „exit()“

```
root@tadas-MS-7984:/home/tadas# drakplayground 0
[2024-05-06 20:37:00,456][INFO] Ensuring that drakrun@0 service is stopped...
[2024-05-06 20:37:00,495][INFO] Generated VM configuration for vm-0
[2024-05-06 20:37:00,525][DEBUG] iptables chains already exist, no setup needed
[2024-05-06 20:37:00,537][INFO] Created bridge drak0
[2024-05-06 20:37:00,577][INFO] Bridge drak0 is up
[2024-05-06 20:37:00,700][INFO] Restoring VM vm-0
Loading new save file /var/lib/drakrun/volumes/snapshot.sav (new xl fmt info 0x3
/0x0/2033)
 Savefile contains xl domain config in JSON format
Parsing config from /etc/drakrun/configs/vm-0.cfg
xc: info: Found x86 HVM domain from Xen 4.16
xc: info: Restoring domain
xc: info: Restore successful
xc: info: XenStore: mfn 0xfeffc, dom 0, evt 1
xc: info: Console: mfn 0xfefff, dom 0, evt 2
Dnsmasq version 2.90  Copyright (c) 2000-2024 Simon Kelley
Compile time options: IPv6 GNU-getopt DBus no-UBus i18n IDN DHCP DHCPv6 no-Lua T
FTP conntrack ipset no-nftset auth cryptohash DNSSEC loop-detect inotify dumpfil
e

This software comes with ABSOLUTELY NO WARRANTY.
Dnsmasq is free software, and you are welcome to redistribute it
under the terms of the GNU General Public License, version 2 or 3.
[2024-05-06 20:37:02,933][DEBUG] Using selector: EpollSelector

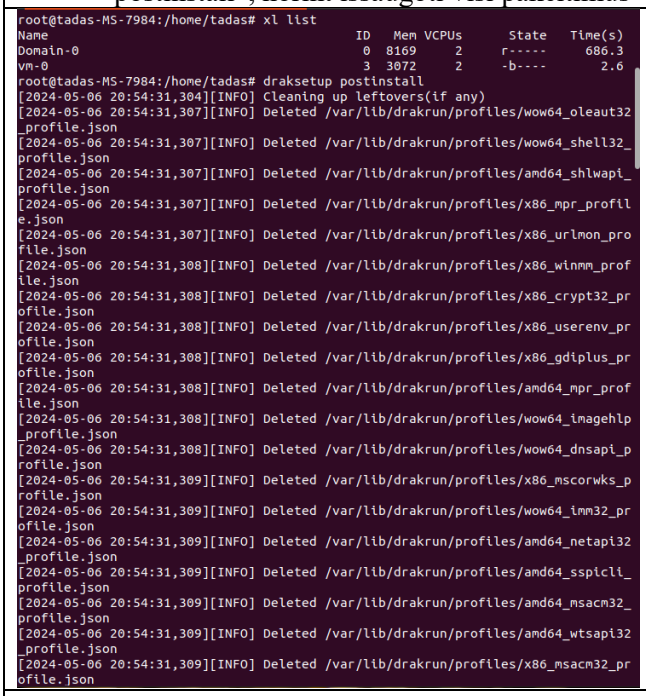
*** Welcome to drakrun playground ***
Your VM is now ready and running with internet connection.
You can connect to it using VNC (password can be found in /etc/drakrun/scripts/c
fg.template)
Run help() to list available commands.

[2024-05-06 20:37:03,024][DEBUG] Using selector: EpollSelector
In [1]: copy('/home/tadas/Documents/npp.8.6.5.Installer.exe')

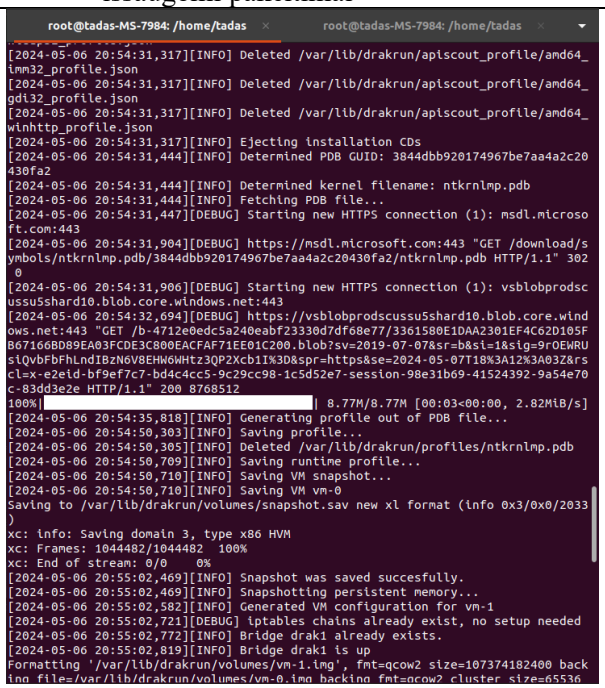
In [2]: exit()

[2024-05-06 20:45:27,109][INFO] Destroying vm-0
[2024-05-06 20:45:29,228][INFO] Bridge drak0 is down
[2024-05-06 20:45:29,295][INFO] Deleted drak0 bridge
root@tadas-MS-7984:/home/tadas# xl list
Name                                        ID   Mem VCPUs      State   Time(s)
Domain-0                                     0  8169     2     r-----     445.7
root@tadas-MS-7984:/home/tadas#
```

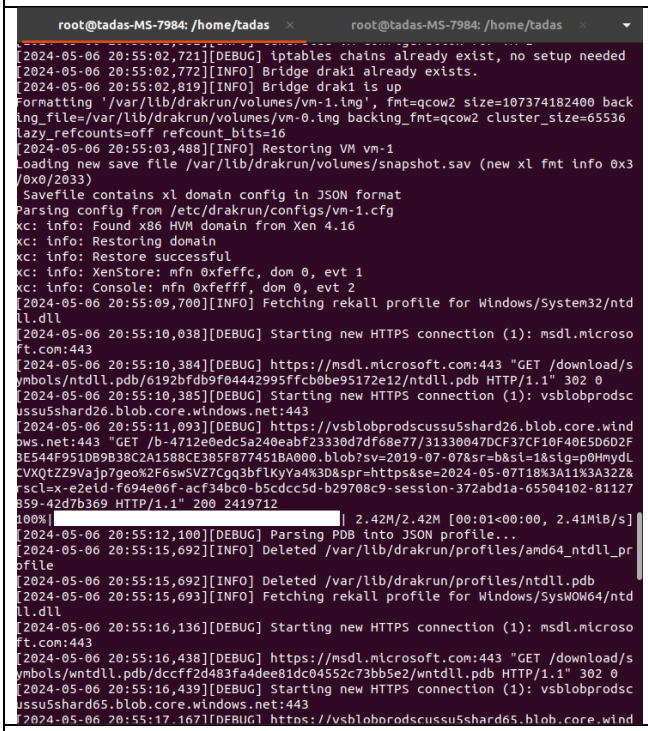
3. Pakartotinai vykdoma „draksetup postinstall“, norint išsaugoti visi pakeitimus



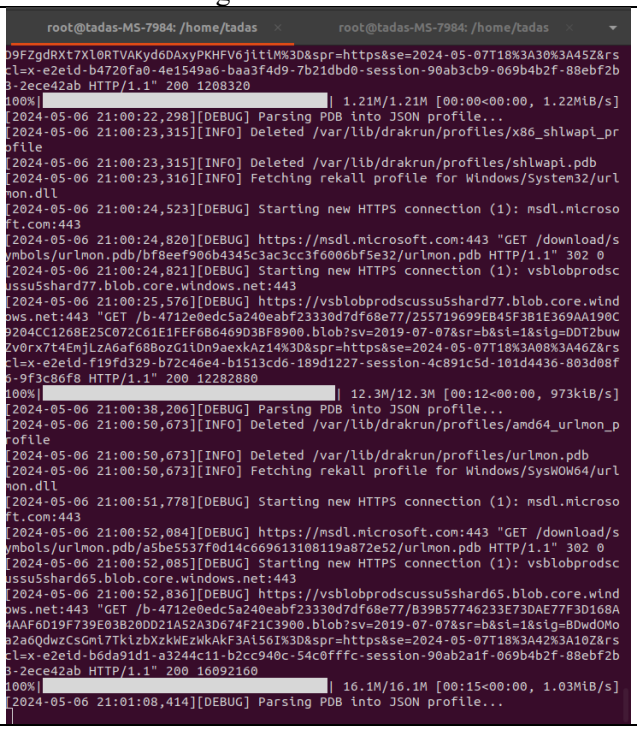
4. Senieji „Rekall“ profiliai ištrinami ir išsaugomi pakeitimai



5. Naujų „Rekall“ profilių formavimo procesas

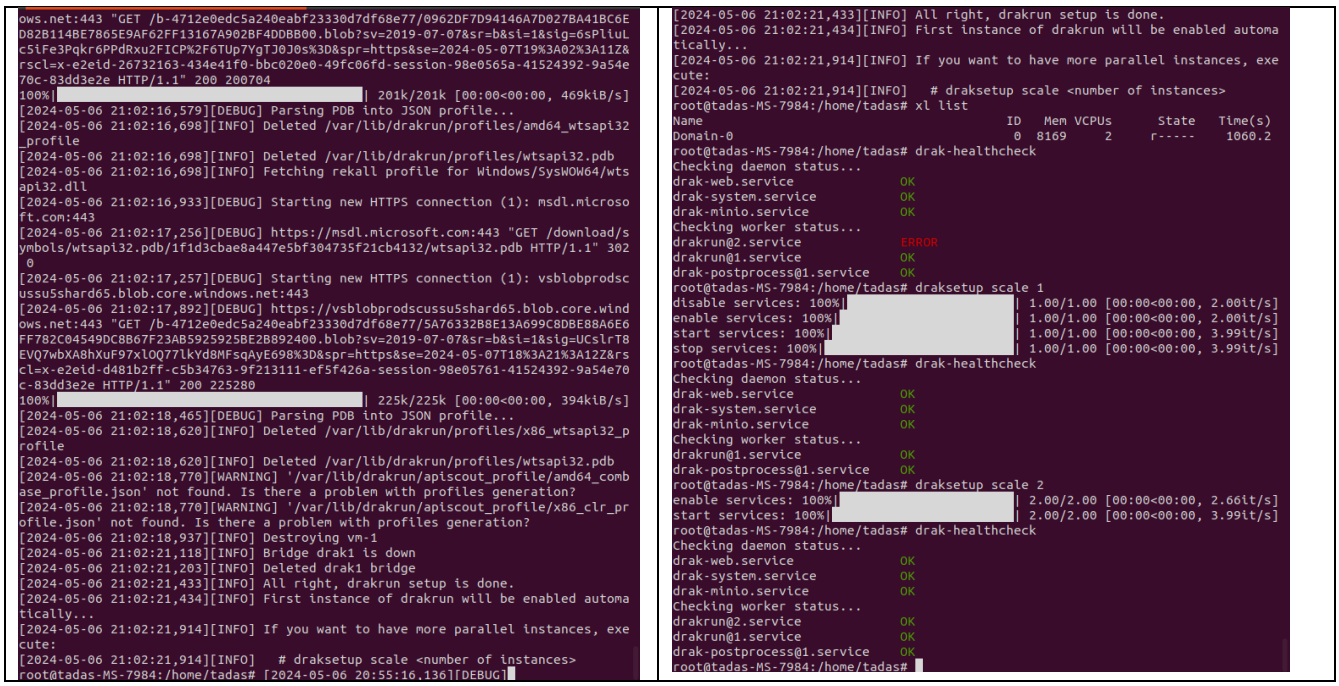


6. Naudojami „Windows“ serveriai reikalingiems failams išgauti

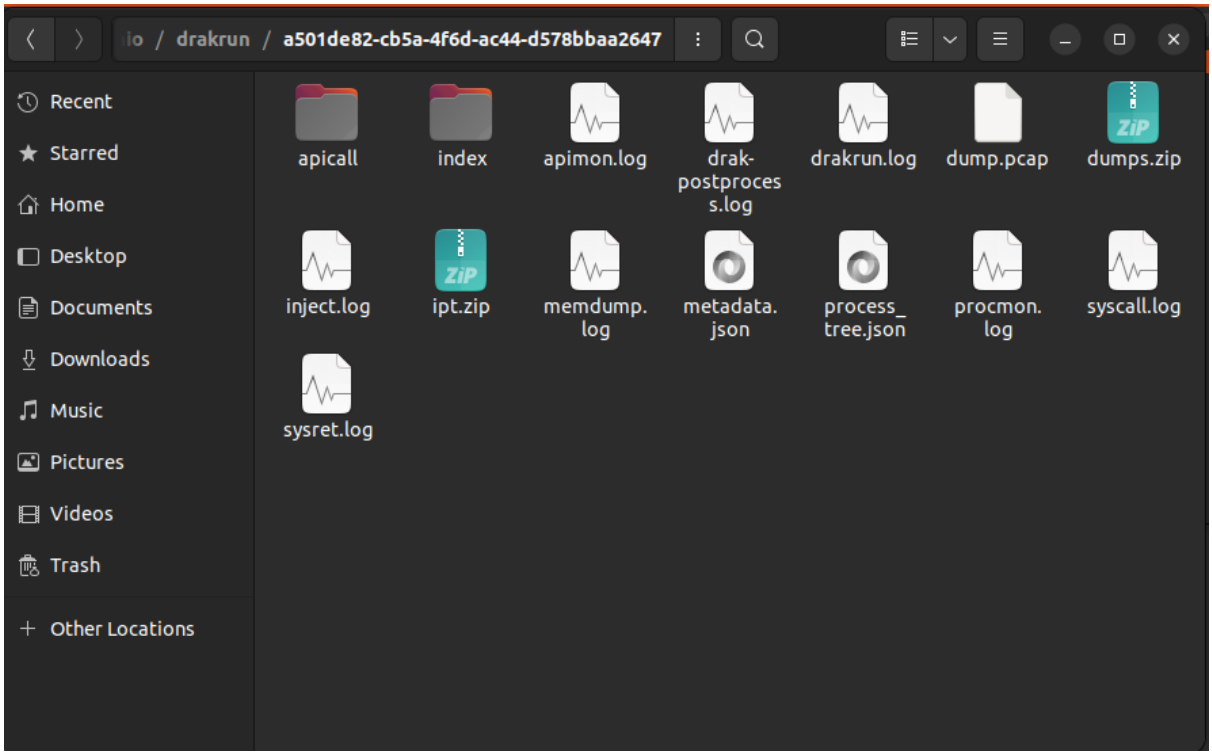


7. Proceso pabaiga

8. Pakartotinai atliekamas paralelių VM konfigūravimas ir paruošimas darbui

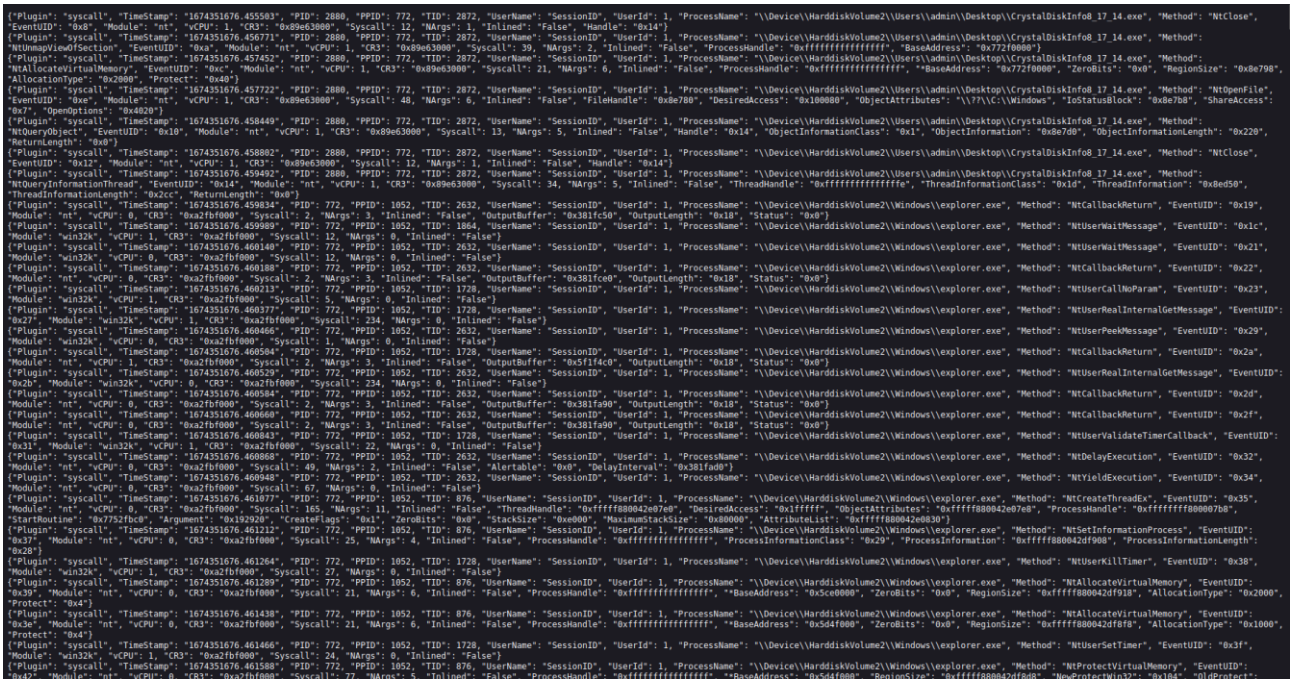


Priedas nr. 3 – „Drakvuf Sandbox“ vieno „Windows“ vykdomojo failo pavyzdžio analizės išvesties failų pavyzdys



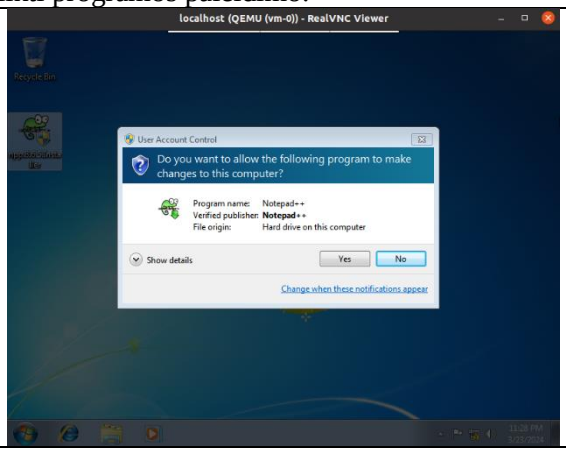
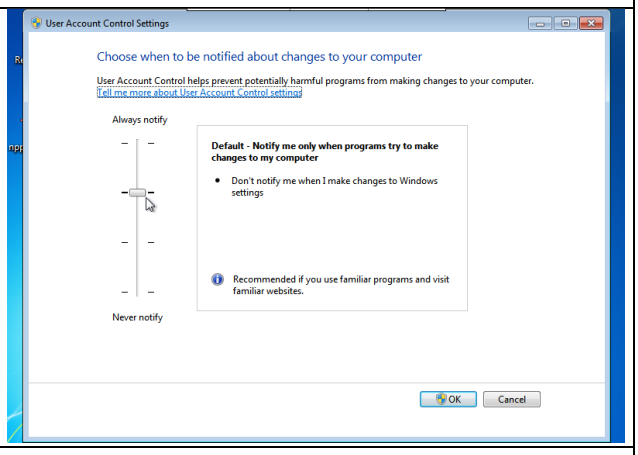
Priedas nr. 4 – „syscall.log“ failo struktūros pavyzdys.

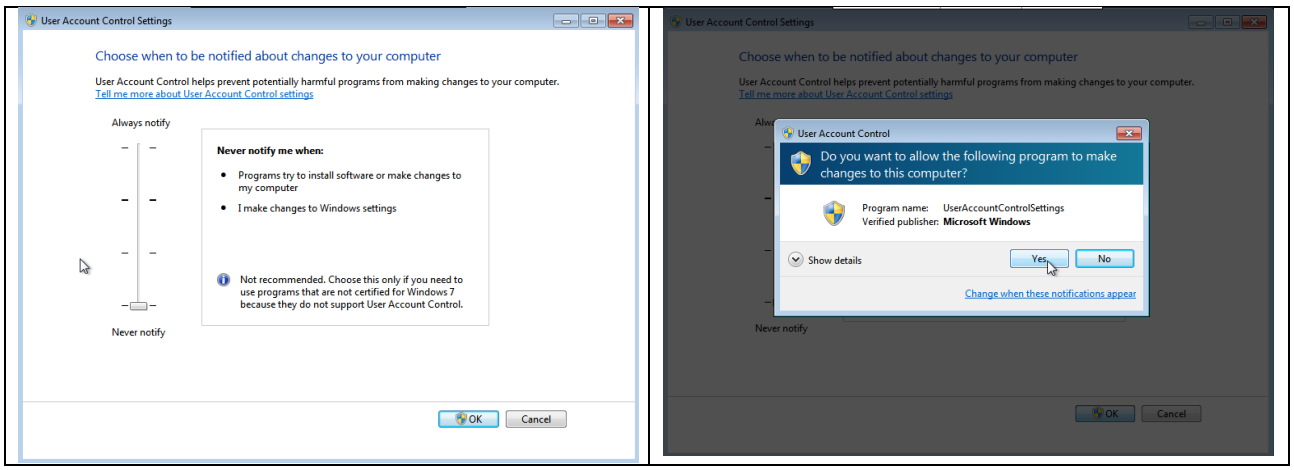
Faile talpinami visi vieno pavyzdžio analizės metu išgauti sisteminiai kreipiniai (nefiltruoti)



Priedas nr. 5 – „Windows 7“ OS „User Account Control Settings“ konfigūravimas

Siekiant išvengti „Drakvuf“ nesklandumų, kuomet tiriamas pavyzdys atsisako būti paleistas, atliekant OS konfigūraciją, būtina naudojant „VNC“ prisijungti prie „Guest“ aplinkos ir atlikti šiuos veiksmus.

<table>
<tr><td>Pranešimas, kuris neleidžia „Drakvuf“ įrankiui atlikti programos paleidimo:</td><td>1. Atidaryti „User Account Control Settings“</td></tr>
<tr><td>
</td><td>
</td></tr>
<tr><td>2. Nustatyti „Never notify“</td><td>3. Patvirtinti nustatymus</td></tr>
</table>



Priedas nr. 6 – Sukurtos „Drak Management“ sistemos demonstracija su 100 KPK pavyzdžių imtimi

Demonstracinė medžiaga gali būti pasiekta per šią nuorodą: https://www.youtube.com/watch?v=PZrxU8YJlkg arba literatūros šaltinių sąraše (Švabauskas, 2024)

Laiko žymės:

- 00:01:47 – tikrinami „Drakvuf“ servisų gyvybingumas
- 00:03:00 – aktyvuojama „Python“ virtuali aplinka
- 00:05:00 – pasiruošiama KPK rūšiavimui, tvarkoma konfigūracija
- 00:05:55 – KPK rūšiavimo proceso paleidimas
- 00:07:00 – tvarkoma „Drak Management“ įrankio konfigūracija
- 00:08:20 – paleidžiamas „Drak Management“ įrankis – tyrimo pradžia
- 00:49:42 – tikrinamas „Xen“ hipervizoriaus statusas
- 01:32:10 – tyrimo pabaiga
- 01:32:40 – gauti tyrimo rezultatų failai
- 01:33:00 – „session“ duomenų bazės peržiūra
- 01:34:00 – „aggregated“ duomenų bazės peržiūra
- 01:34:45 – „database“ duomenų bazės peržiūra
- 01:37:21 – sisteminių kreipinių filtravimas: išskirtiniai procesų pavadinimai
- 01:40:06 – komandinės eilutės išvesties failas

Priedas nr. 7 – 1000 KPK tyrimo rezultatų failai

Galutinio empirinio tyrimo etapo gauti rezultatų failai yra pateikiami „Google Drive“ debesyje (Švabauskas, 2024). Pateikiamų failų paaiškinimai (tik svarbūs):

- „aggregated.db“ – „SQLite“ duomenų bazės failas, kuriame saugomi visi agreguoti sisteminiai kreipiniai po atlikto tyrimo.
- „console_output.log“ – tyrimo vykdymo metu komandinės eilutės–*konsolės* žurnalo kopijos failas.
- „database.db“ – „SQLite“ duomenų bazės failas, kuriame saugomi išfiltruoti visų KPK pavyzdžių visi sisteminiai kreipiniai.
- „session.db“ – „SQLite“ duomenų bazės failas, kuriame saugomi visų KPK pavyzdžių analizių statistinių rezultatų suvestinės.

Priedas nr. 8 – Dukteriniai procesai, unikalūs jų pavadinimai gauti atlikus tyrimą su 1000 KPK

```
1A0D0C0C120D15
1A0F0F0D120D15
1B0F0C0E120A15
21F1.tmp
2348.tmp
27EA.tmp
4877b76d30603e
7D57AD13E21.ex
B36.tmp
CTS.exe
HelpMe.exe
Isass.exe
LSASSMGR.EXE
Logo1_.exe
NJLSJGSRYN.exe
Scegli_nome_al
Unicorn-11717.
Unicorn-12026.
Unicorn-12293.
Unicorn-12826.
Unicorn-14216.
Unicorn-1451.e
Unicorn-14514.
Unicorn-16570.
Unicorn-16954.
Unicorn-17533.
Unicorn-19367.
Unicorn-19886.
Unicorn-24162.
Unicorn-24695.
Unicorn-26440.
Unicorn-2646.e
Unicorn-2864.e
Unicorn-2973.e
Unicorn-29986.
Unicorn-31562.
Unicorn-32151.
Unicorn-33674.
Unicorn-35838.
Unicorn-3741.e
Unicorn-37491.
Unicorn-38495.
Unicorn-40358.
Unicorn-40690.
```

```
Unicorn-4092.e
Unicorn-4125.e
Unicorn-41650.
Unicorn-41842.
Unicorn-42937.
Unicorn-4317.e
Unicorn-43487.
Unicorn-43814.
Unicorn-44424.
Unicorn-44582.
Unicorn-44774.
Unicorn-45708.
Unicorn-45926.
Unicorn-45992.
Unicorn-46239.
Unicorn-46568.
Unicorn-48168.
Unicorn-49050.
Unicorn-49167.
Unicorn-49434.
Unicorn-51650.
Unicorn-53134.
Unicorn-53158.
Unicorn-53283.
Unicorn-53326.
Unicorn-54089.
Unicorn-55355.
Unicorn-56258.
Unicorn-56891.
Unicorn-57026.
Unicorn-61124.
Unicorn-61297.
Unicorn-61302.
Unicorn-61494.
Unicorn-61551.
Unicorn-61873.
Unicorn-63357.
Unicorn-63910.
Unicorn-64056.
Unicorn-65002.
Unicorn-65386.
Unicorn-6541.e
Unicorn-6922.e
Unicorn-7057.e
Unicorn-7436.e
Unicorn-7825.e
Unicorn-8017.e
Unicorn-809.ex
Unicorn-8209.e
Unicorn-8785.e
Unicorn-8922.e
Unicorn-8977.e
Unicorn-9619.e
VI21B2.tmp
VirusShare_00c
VirusShare_29f
VirusShare_3bc
VirusShare_523
VirusShare_570
VirusShare_578
VirusShare_6e1
VirusShare_6f7
VirusShare_7f0
VirusShare_8f1
VirusShare_bad
VirusShare_c45
WMIC.exe
WbX4KHzLDp52JW
WwanSvc.exe
XHCQFIVYND.exe
\Device\HarddiskVolume2\Program Files (x86)\4877b7a4\jusched.exe
\Device\HarddiskVolume2\Program Files (x86)\avesfnme\zccml.exe
\Device\HarddiskVolume2\Program Files (x86)\bhcz\copvvtyjpfenkn.exe
\Device\HarddiskVolume2\Program Files (x86)\bmuk\tacre.exe
\Device\HarddiskVolume2\Program Files (x86)\jcctrjauo\niwtgsgiq.exe
\Device\HarddiskVolume2\Program Files (x86)\jhiby\epyrrcixryyix.exe
\Device\HarddiskVolume2\Program Files (x86)\jqcsqq\ptbszjqgukvn.exe
\Device\HarddiskVolume2\Program Files (x86)\tun\bzfl.exe
\Device\HarddiskVolume2\Program Files (x86)\uitylmep\lou.exe
\Device\HarddiskVolume2\Program Files (x86)\vifw\wpujcvnd.exe
```

```
\Device\HarddiskVolume2\Program Files (x86)\vsztrrfyy\diqemktywdnf.exe
\Device\HarddiskVolume2\Program Files (x86)\weom\q.exe
\Device\HarddiskVolume2\Program Files (x86)\wsndauymxx\nbjdxqvkjh.exe
\Device\HarddiskVolume2\Program Files (x86)\ydymza\nkfwaqbxrgaoy.exe
\Device\HarddiskVolume2\Program Files (x86)\zoxkxq\bqkldalznmwk.exe
\Device\HarddiskVolume2\ProgramData\NSIEMgMQ\kygEYUwY.exe
\Device\HarddiskVolume2\ProgramData\Update\WwanSvc.exe
\Device\HarddiskVolume2\Users\Public\Microsoft Build\Isass.exe
\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\21F1.tmp
\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\2348.tmp
\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\27EA.tmp
\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\40A7.tmp
\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\42BA.tmp
\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\4877b76d30603ec4.exe
\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\B36.tmp
\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\B7KVnaUCjmbQpXd.exe
\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\VI21B2.tmp
\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\WbX4KHzLDp52JWS.exe
\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\biudfw.exe
\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\budha.exe
\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\codecupdate.exe
\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\edkugeo.exe
\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\hcbnaf.exe
\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\huter.exe
\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\lasma.exe
\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\quip.exe
\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\quity.exe
\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\sander.exe
\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\uImjJGvK4kLFwah.exe
\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\v96GeueVzFPLBvb.exe
\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\virusshare 93a66ded810190c55b7d85883b13890a\vir
usshare_93a66ded810190c55b7d85883b13890a.exe
\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\wytyg.exe
\Device\HarddiskVolume2\Users\box\AppData\Local\Temp\~BF98E.tmp
\Device\HarddiskVolume2\Users\box\AppData\Roaming\7D57AD13E21.exe
\Device\HarddiskVolume2\Users\box\AppData\Roaming\Scegli_nome_allegato.exe
\Device\HarddiskVolume2\Users\box\AppData\Roaming\facebook update.exe
\Device\HarddiskVolume2\Users\box\Desktop\1A0D0C0C120D156B155B15A0A0B160B0F160E.exe
\Device\HarddiskVolume2\Users\box\Desktop\1A0F0F0D120D156D155A15F0F0E160B0D160C.exe
\Device\HarddiskVolume2\Users\box\Desktop\1B0F0C0E120A156D155C15D0D0B160F0E160E.exe
\Device\HarddiskVolume2\Users\box\Desktop\NJLSJGSRYN.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-11717.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-12026.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-12293.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-12826.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-14216.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-1451.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-14514.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-16570.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-16954.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-17533.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-19367.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-19580.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-19886.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-24162.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-24695.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-26440.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-2646.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-2864.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-2973.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-29986.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-31562.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-32151.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-33674.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-35838.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-3741.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-37491.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-38495.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-40358.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-40690.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-4092.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-4125.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-41650.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-41842.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-42937.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-4317.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-43487.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-43814.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-44424.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-44582.exe
```

```
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-44774.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-45708.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-45926.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-45992.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-46239.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-46568.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-48168.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-49050.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-49167.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-49434.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-51650.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-53134.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-53158.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-53283.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-53326.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-55355.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-56258.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-56891.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-57026.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-61124.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-61297.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-61302.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-61494.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-61551.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-61873.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-63357.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-63910.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-64056.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-65002.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-65386.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-6541.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-6922.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-7057.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-7436.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-7825.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-8017.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-809.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-8209.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-8785.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-8922.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-8977.exe
\Device\HarddiskVolume2\Users\box\Desktop\Unicorn-9619.exe
\Device\HarddiskVolume2\Users\box\Desktop\XHCQFIVYND.exe
\Device\HarddiskVolume2\Users\box\peuineb.exe
\Device\HarddiskVolume2\Users\box\syoYEkso\oMcwMcQI.exe
\Device\HarddiskVolume2\Windows\CTS.exe
\Device\HarddiskVolume2\Windows\Logo1_.exe
\Device\HarddiskVolume2\Windows\Microsoft.NET\Framework\v2.0.50727\dw20.exe
\Device\HarddiskVolume2\Windows\SysWOW64\HelpMe.exe
\Device\HarddiskVolume2\Windows\SysWOW64\LSASSMGR.EXE
\Device\HarddiskVolume2\Windows\SysWOW64\cmd.exe
\Device\HarddiskVolume2\Windows\SysWOW64\ctfmon.exe
\Device\HarddiskVolume2\Windows\SysWOW64\drivers\winlogon.exe
\Device\HarddiskVolume2\Windows\SysWOW64\explorer.exe
\Device\HarddiskVolume2\Windows\SysWOW64\lssmon.exe
\Device\HarddiskVolume2\Windows\SysWOW64\netsh.exe
\Device\HarddiskVolume2\Windows\SysWOW64\reg.exe
\Device\HarddiskVolume2\Windows\SysWOW64\sc.exe
\Device\HarddiskVolume2\Windows\SysWOW64\schtasks.exe
\Device\HarddiskVolume2\Windows\SysWOW64\srtsrv32.exe
\Device\HarddiskVolume2\Windows\SysWOW64\svchost.exe
\Device\HarddiskVolume2\Windows\SysWOW64\winver.exe
\Device\HarddiskVolume2\Windows\System32\bcdedit.exe
\Device\HarddiskVolume2\Windows\System32\cmd.exe
\Device\HarddiskVolume2\Windows\System32\conhost.exe
\Device\HarddiskVolume2\Windows\System32\net.exe
\Device\HarddiskVolume2\Windows\System32\reg.exe
\Device\HarddiskVolume2\Windows\System32\sc.exe
\Device\HarddiskVolume2\Windows\System32\schtasks.exe
\Device\HarddiskVolume2\Windows\System32\vssadmin.exe
\Device\HarddiskVolume2\Windows\System32\wbem\WMIC.exe
\Device\HarddiskVolume2\Windows\System32\wermgr.exe
\Device\HarddiskVolume2\Windows\System32\wevtutil.exe
\Device\HarddiskVolume2\Windows\alg.exe
\Device\HarddiskVolume2\Windows\outlook.exe
\Device\HarddiskVolume2\_v_5539.exe
\Device\HarddiskVolume2\exc.exe
_v_5539.exe
alg.exe
bcdedit.exe
```

```
biudfw.exe
budha.exe
bzfl.exe
cmd.exe
codecupdate.ex
conhost.exe
copvvtyjpfenkn
ctfmon.exe
dw20.exe
edkugeo.exe
exc.exe
explorer.exe
facebook updat
hcbnaf.exe
huter.exe
jusched.exe
kygEYUwY.exe
lasma.exe
lou.exe
lssmon.exe
nbjdxqvkjh.exe
net.exe
netsh.exe
niwtgsgiq.exe
nkfwaqbxrgaoy.
oMcwMcQI.exe
outlook.exe
peuineb.exe
q.exe
quip.exe
quity.exe
reg.exe
sander.exe
sc.exe
schtasks.exe
srtsrv32.exe
tacre.exe
virusshare_93a
vssadmin.exe
wermgr.exe
wevtutil.exe
winlogon.exe
winver.exe
wpujcvnd.exe
wytyg.exe
```